بسم الله الرحمن الرحيم

وزارت ارتباطات و فناوری اطلاعات
معاونت فناوری اطلاعات

راهنمای نرم‌افزارهای سازمانی آزاد/ متن باز
(سیستم عامل)

طرح تدوین طرح جامعه فناوری اطلاعات کشور
(کد پروژه:ITMP-40812001-04-3-4)

مجری طرح:	مهندس عبدالمجید ریاضی، معاون فناوری اطلاعات وزارت ارتباطات و فناوری اطلاعات
ناظر طرح:	دکتر علی ناصری، عضو هیئت علمی دانشگاه امام حسین(ع) و مدیر کل امور زیر بنایی فناوری اطلاعات وزارت ارتباطات و فناوری اطلاعات
مجری پروژه:	مرکز تحقیقات مخابرات ایران
ناظر پروژه:	دکتر حمیدرضا ربیعی، عضو هیئت علمی دانشگاه صنعتی شریف
مؤلفان:	بهروز رحمتی، مهدی امیری کردستانی، حامد آبانگر

سال ۱۳۸۷

سرشناسه:	رحمتی، بهروز، ۱۳۵۵
عنوان و نام پدیدآور:	راهنمای نرم افزارهای سازمانی آزاد/متن باز(سیستم عامل): طرح تدوین طرح جامع فناوری اطلاعات کشور (ITMP-40812001-04-3-4) ؛ مؤلفان بهروز رحمتی، مهدی امیری کردستانی، حامد آبانگر؛ مجری طرح عبدالمجید ریاضی؛ ناظر طرح علی ناصری؛ ناظر پروژه حمید رضا ربیعی؛ مجری پروژه مرکز تحقیقات مخابرات ایران؛ [برای وزارت ارتباطات و فناوری اطلاعات].
مشخصات نشر:	تهران: جهان جام جم، ۱۳۸۷.
مشخصات ظاهری:	۲۰۸ ص. مصور، جدول.
شابک:	978-964-8625-875
وضعیت فهرست نویسی:	فیپا
یادداشت:	کتابنامه : ص [۲۰۷-۲۰۸]
موضوع:	نرم افزار متن باز.
موضوع:	سیستم عامل (کامپیوتر)
موضوع:	بازرگانی، برنامه های کامپیوتری
شناسه افزوده:	امیری کردستانی، مهدی، ۱۳۵۹
شناسه افزوده:	آبانگر، حامد،۱۳۵۹
شناسه افزوده:	ریاضی، عبدالمجید، مجری طرح
شناسه افزوده:	ناصری، علی، ۱۳۴۸
شناسه افزوده:	ربیعی، حمیدرضا
شناسه افزوده:	مرکز تحقیقات مخابرات ایران.
شناسه افزوده:	ایران. وزارت ارتباطات و فناوری اطلاعات. معاونت فناوری اطلاعات.
رده بندی کنگره:	QA۷۶/۷۶/آ۴ ر۳ ۱۳۸۷
رده بندی دیویی:	۰۰۵/۲۶
شماره کتابشناسی ملی:	۱۲۱۰۰۷۷

راهنمای نرم‌افزارهای سازمانی آزاد/ متن باز (سیستم عامل)

مؤلفان: بهروز رحمتی، مهدی امیری کردستانی، حامد آبانگر

حروف چینی و صفحه‌بندی: زهرا محبی متین

ویراستار محتوایی: محمدمهدی نظام آبادی

ویراستار نهایی : لیلا بیگ

مسئول فنی: عاطفه قوامی‌فر

گرافیست: ناصر اسماعیلی

نوبت چاپ: اول

شمارگان: ۲۰۰۰

سال انتشار: ۱۳۸۷

ناشر: موسسه انتشارات جهان جام جم

نشانی: تهران- جنت آباد- خ گلزار غربی-کوچه شهید محسنی بعد- پلاک ۱۳- تلفکس: ۴۴۸۲۹۲۳۱

پیشگفتار

زندگی بشر از عصر تولید انبوه به عصر ارتباطات و اطلاعات ارتقاء یافته و حرکت تکاملی کشورهای جهان به سوی جوامع اطلاعاتی و دانش بنیان، کلیه فرایندها و فعالیت‌های اقتصادی، فرهنگی، صنعتی، سیاسی و روابط اجتماعی را تحت تاثیر قرار داده است.

چارچوب ساختاری تشکیل دهنده این عصر را تولید، پردازش، انتقال و مدیریت اطلاعات و ارتباطات به منظور ایجاد پایگاه‌های دانش و معرفت فردی، گروهی، سازمانی و کشور تشکیل می‌دهد و لذا فناوری اطلاعات را که شامل فناوری‌های بکارگرفته شده در فرایند مذکور می‌باشد برای جوامع بشری به عنوان عامل حیاتی و تعیین کننده مطرح ساخته است.

در دنیای امروز اطلاعات نه تنها به عنوان یکی از منابع و دارایی‌های اصلی سازمان‌ها شناخته می‌شود بلکه درحکم وسیله و ابزاری برای مدیریت اثر بخش بر سایر منابع و دارایی‌های سازمان (منابع مالی، نیروی انسانی وغیره) نیز محسوب می‌شود و لذا از اهمیت و ارزش ویژه‌ای برخوردار گشته است. اما این ارزش تنها در صورتی محقق و دست یافتنی خواهد بود که اطلاعات بتوانند در زمان مناسب، با کیفیت مطلوب و امنیت قابل قبول در اختیار افراد مناسب قرارگیرد و ارتباطات به صورت مطلوب و بهینه در سازمان برقرار گردد. از این رو است که فناوری اطلاعات که زمینه‌ساز انتقال، جابجایی، بکارگیری و مدیریت موثر اطلاعات در کشورها می‌باشد، از اهمیتی حیاتی برخوردار گشته است.لذا در راستای چشم‌انداز بیست ساله جمهوری اسلامی ایران مبنی بر تحقق جامعه‌ای توسعه یافته، متناسب با مقتضیات فرهنگی، متکی بر اصول اخلاقی و ارزش‌های اسلامی، حفظ هویت ایرانی اسلامی با تاکید بر مردم سالاری دینی و عدالت اجتماعی، آزادی‌های مشروع، حفظ کرامت و حقوق انسان‌ها و برخوردار از دانش پیشرفته از یک طرف، و تاثیر عمیق فناوری اطلاعات و ارتباطات بر ابعاد مختلف زندگی بشر و بالاخص نقش حیاتی و حساس آن درجنبه‌های فرهنگی، اقتصادی، امنیتی، اجتماعی و سیاسی از طرف دیگر وزارت ارتباطات و فناوری اطلاعات را برآن داشت که جهت برنامه‌ریزی کلان توسعه فناوری اطلاعات اقدام به پیشنهاد طرح تدوین طرح جامع فناوری اطلاعات کشور به سازمان مدیریت و برنامه‌ریزی نماید که بعد از بررسی کارشناسی طی موافقتنامه‌ای به امضاء طرفین (وزارت ارتباطات و فناوری اطلاعات و سازمان مدیریت و برنامه‌ریزی) رسید این موافقتنامه شامل ۵ پروژه اصلی؛ تدوین طرح کلان فناوری اطلاعات، تدوین برنامه اجرایی وظایف وزارت در حوزه امنیت (افتا)، ایجاد بانک اطلاعاتی وضعیت فناوری

اطلاعات، تدوین چارچوب کاربردهای فناوری اطلاعات در کشور، تهیه و پیش‌نویس لوایح و مقررات حقوقی است که هر کدام حاوی زیر بخش‌های مختلف می‌باشند.

امید است انجام فعالیت‌های مندرج در این طرح بتواند زمینه توسعه و ارتقاء صنعت و خدمات فناوری اطلاعات در کشور را مهیا سازد. این کتاب نتیجه حاصل یکی از فعالیت‌های طرح می‌باشد که با همت کارشناسان فناوری اطلاعات به سرانجام رسیده است.

عبدالمجید ریاضی

مجری طرح تدوین طرح جامع فناوری اطلاعات

مقدمه

نفوذ فناوری اطلاعات در حوزه‌های مختلف تحول عظیمی در نحوه زندگی بشر ایجاد نموده است. بسترها و ابزارهای لازم برای ورود فناوری اطلاعات به عرصه‌های گوناگون نقش تعیین کننده‌ای در میزان اثر بخشی این فناوری در آن عرصه ایفا می‌کنند. زیر ساخت‌های ارتباطی، سیستم‌های سخت افزاری، نرم‌افزارهای مختلف و تجهیزات امنیتی همه و همه از جمله الزامات استفاده از فناوری اطلاعات به شمار می‌روند. همه گیر شدن استفاده از این فناوری از یک طرف و ارئه ابزارهای عملیاتی گوناگون توسط محققین و برنامه نویسان از طرف دیگر لزوم ارائه ابزارهای یکپارچه و در اصطلاح در سطح سازمان را آشکار ساخته است. این مسئله سبب گردیده است تا مجموعه‌های استفاده کننده از فناوری اطلاعات برای رسیدن به راندمان بالاتر و ارائه خدمات با کیفیت تر عمدتاً در پی راهکارهای یکپارچه و در سطح سازمان باشند.

سیستم عامل با توجه به دارا بودن نقش کلیدی در هر سیستم رایانه‌ای از اهمیت ویژه‌ای برخوردار است و انواع متن‌باز آن با توجه به ویژگی‌ها و قابلیت های منحصر به فرد مورد توجه بسیاری از محققین قرارگرفته است. هدف اصلی کتاب حاضر ارتقاء سطح دانش موجود در کشور در زمینه نرم‌افزارهای آزاد/ متن‌باز بوده است که با توجه به روند جهانی در این حوزه لازم است مورد توجه بیشتری قرارگیرد.

در فصل اول کتاب مشخصات و انواع مختلف سیستم‌عامل‌ها مورد بررسی قرارگرفته و نحوه عملکرد آنها مطالعه گردیده است. در فصل دوم در ابتدا دسته بندی سیستم‌عامل‌های متن‌باز انجام شده و در ادامه انواع مختلف سیستم‌عامل‌های متن باز معرفی و ویژگی‌های هریک بیان شده است. این فصل با مقایسه انواع سیستم‌عامل‌های متن باز از جنبه‌های گوناگون پایان می‌پذیرد. در فصل سوم تغییرات لازم برای بومی‌سازی سیستم‌عامل لینوکس بررسی گردیده‌اند. فصل چهارم به طور خاص به موضوع سیستم‌عامل‌های بلادرنگ و تعبیه شده پرداخته و مؤلفه‌های مختلف این نوع از سیستم‌عامل‌ها را مورد مطالعه قرار می‌دهد. در فصل پنجم انواع مختلف سیستم‌عامل‌های بلادرنگ و تعبیه شده متن باز معرفی و از لحاظ ساختار داخلی و کاربردهای مناسب بررسی می‌شوند. سه فصل آخر کتاب به موضوع سیستم‌عامل‌های کارت‌های هوشمند پرداخته است. بدین صورت که در ابتدا انواع مختلف کارت‌های هوشمند، استانداردها و سیستم‌عامل‌های آنها مورد بررسی قرار می‌گیرند. در ادامه سیستم‌عامل‌های متن‌باز مربوط به این کارت‌ها نظیر Multos و چند نمونه دیگر مورد بررسی قرارگرفته و ویژگی‌ها و قابلیت‌های هریک بیان گردیده است. در فصل آخر تغییرات لازم جهت بومی سازی سیستم عامل کارت هوشمند شناسایی و بررسی شده و اهمیت، دلایل و ابزارهای لازم برای انجام این کار ارائه شده است.

امید است کتاب حاضر با توجه به اهمیت بحث نرم‌افزارهای آزاد / متن‌باز بتواند نقشی هرچند اندک در ارتقاء سطح دانش این حوزه در کشور داشته باشد.

فهرست مطالب

عنوان **صفحه**

فهرست مطالب

عنوان **صفحه**

فهرست مطالب

عنوان **صفحه**

فهرست مطالب

عنوان **صفحه**

فهرست مطالب

عنوان **صفحه**

فهرست مطالب

عنوان **صفحه**

فهرست مطالب

عنوان **صفحه**

فصل اول – مشخصات سیستم عامل‌ها

امروزه کامپیوتر بدون برنامه‌های آن جز پاره‌ای از آهن آلات بدون مصرف چیزی نیست، کامپیوترها با برنامه‌های آن، تبدیل به ابزاری پر مصرف برای ذخیره و بازیابی اطلاعات، پردازش و آمار، پخش صدا و تصویر، ارسال نامه، جستجو در اینترنت و بسیاری از کاربردهای با ارزش می‌شود.

بطور کلی، برنامه‌های کامپیوتر را به دو دسته برنامه‌های سیستمی و برنامه‌های کاربردی تقسیم می‌کنند. برنامه‌های سیستمی برنامه‌های هستند که خود کامپیوتر و عملکرد آن را مدیریت و کنترل می‌کنند در حالیکه برنامه کاربردی در اصل نیاز کاربر را برآورده می سازند و کاری را انجام می‌دهند که کاربر انتظار دارد.

بنیادی ترین برنامه سیستمی، سیستم عامل نام دارد و وظیفه آن مدیریت منابع کامپیوتر و به وجود آوردن لایه ای بر روی سخت‌افزار کامپیوتر است که برنامه های کاربردی بر روی آن نوشته می شوند و قادرند در آن اجرا شوند.

کامپیوتر های امروزی متشکل از یک یا چند پردازنده، مقداری حافظه اصلی، چندین دیسک، صفحه کلید، چاپگر، یک مانیتور و اتصالات شبکه و بسیاری دستگاه های ورودی/ خروجی دیگر هستند که در کل یک سیستم پیچیده و کامل به وجود می آورند. نوشتن برنامه‌هایی که بتواند از این ابزارها بطور صحیح استفاده کند کار ساده ای نیست و اگر قرار بود هر برنامه نویس نگران طریقه عملکرد دیسک سخت و دو جین خطاهایی که ممکن است به وجود آید بود، هیچگاه این میزان برنامه به وجود نمی آمد. این پیچیدگی سخت‌افزارها سبب شده تا سالها پیش، راه حلی به وجود آید تا برنامه های کاربردی از پیچیدگی های سخت‌افزاری دور نگاه داشته شوند. این راه حل به وجود آمدن یک لایه نرم‌افزاری بر روی سخت‌افزار است که با مدیریت کامل سخت‌افزار یک مدل کار آمد نرم‌افزاری و صد البته ساده تر به برنامه سازان ارائه می‌کند. این مدل کار آمد یا ماشین مجازی ارائه شده همان سیستم عامل است.

۱–۱– سیستم عامل چیست؟

همه کاربران کامپیوتر دید کلی نسبت به سیستم عامل و ماهیت آن دارند ولی ارائه یک تعریف صحیح و جامع از سیستم عامل کمی مشکل به نظر می رسد. سیستم عامل دو نقش

پایه ای بسیار مهم؛ ۱- بسط دادن ماشین و ۲- مدیریت منابع را ایفا می‌کند و همه تعاریف سیستم عامل به یکی از این دو نقش اشاره می‌کنند. اکنون به بررسی این دو نقش می پردازیم:

همانطور که در مقدمه اشاره شد، سیستم عامل جزئیات ریز و پیچیده سخت‌افزار ماشین را به یک مدل نرم‌افزاری مناسب یا یک ماشین مجازی و ساده تبدیل می کند. برنامه سازان به جای فراگیری جزئیات سخت‌افزارهای مختلف و طریقه عملکرد آنها، برای یک ماشین مجازی بسیار ساده تر برنامه خود را می نویسند و از جزئیات سخت‌افزاری کامپیوترهای مختلف صرف نظر می‌کنند.

در یک کامپیوتر مدرن وظیفه سیستم عامل تخصیص منابع سیستم مانند پروسسور، حافظه، چاپگر و غیره به برنامه‌های مختلف و کنترل این منابع است. بسیاری از منابع سیستم مانند چاپگر و پردازنده نمی توانند در آن واحد در اختیار دو برنامه قرار گیرند و وظیفه سیستم عامل تخصیص صحیح این منابع به برنامه های کاربردی است.

۱-۲- تاریخچه سیستم عامل

کامپیوترها و سیستم عامل‌ها طی سالیان گذشته سیر رشد سریعی داشته‌اند. بررسی تاریخچه سیستم عامل، ما را با محورها و خصوصیاتی که از یک سیستم عامل باید توقع داشت، بیشتر آشنا می کند. برای بررسی رشد سیستم‌عامل و تاریخچه آن می باید ابتدا تاریخچه کامپیوتر و رشد آن را بررسی کرد.

۱-۲-۱-کامپیوترهای نسل اول (۱۹۴۵ الی ۱۹۵۵)

کامپیوتر های نسل اول در اواسط دهه ۱۹۴۰، در زمان جنگ جهانی دوم، توسط گروه‌های مختلفی در آمریکا و آلمان به وجود آمدند. این کامپیوترها از لامپ‌های خلاء[1] و تجهیزات الکترونیکی ساده تشکیل شده بودند و حجمی به اندازه یک سالن بزرگ را اشغال می‌کرده اند. در این کامپیوترها، برنامه‌ها توسط بردهای الکترونیکی[2] به کامپیوتر ارائه می شد و برنامه سازان و تولید کنندگان کامپیوتر، همگی در یک تیم بودند.

---

1 Vacuum Tubes
2 Plug Boards

تا اوایل دهه ۱۹۵۰ این کامپیوترها کاملتر شدند و برنامه های آنها توسط کارت‌های پانچ[1] به جای بردهای الکترونیکی به آنها ارائه می‌شد.

۱-۲-۲- کامپیوترهای نسل دوم (۱۹۵۵ الی ۱۹۶۵)

با اختراع ترانزیستورها در اواسط دهه ۱۹۵۰، تصویر کامپیوتر تغیرات بنیادی پیدا کرد. در این زمان کامپیوترها آنقدر پایدار شده بودند تا سازندگان بتوانند آنان را بفروش برسانند و برای اولین بار بین طراحان سیستم، تولید کنندگان، اپراتورها، برنامه سازان و مسئولان پشتیبانی تمایزهای قابل مشاهده ای به وجود آمد.

این کامپیوترها که امروزه ابررایانه نامیده می‌شوند، در اتاقهای مخصوص حفاظت شده نگهداری و توسط اپراتورهای حرفه ای مدیریت و کنترل می‌شده اند. فقط شرکتهای بسیار بزرگ، سازمان‌های دولتی و دانشگاه‌ها توانایی خرید چنین کامپیوتر های چند میلیون دلاری را داشتند.

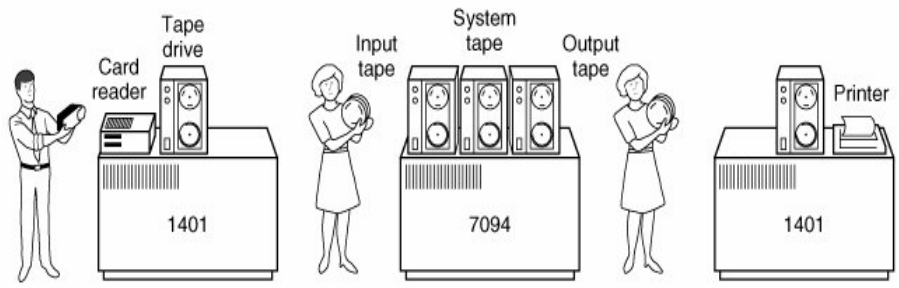
برای اجرای یک برنامه در این کامپیوترها ، برنامه نویسان ابتدا برنامه خود را به زبان فرترن یا اسمبلی نوشته و توسط کارت پانچ به مسئول اتاق ورودی اطلاعات ارائه می کرده اند و سپس بعد از چند ساعت از مسئول اتاق خروجی اطلاعات نتیجه را چاپ شده دریافت می‌کرده اند.

عمده زمان این کامپیوترها در عملیات ورودی / خروجی برای دریافت کارت‌های پانچ و چاپ نتایج صرف می‌شد که مقرون و به صرفه نبود. از این رو طراحان سیستم، کامپیوترهای ارزان قیمتی (مانند ۱۴۰۱) برای عملیات دریافت اطلاعات ورودی و ذخیره بر روی نوار و همچنین چاپ خروجی‌ها از نوار ساختند و کامپیوترهای اصلی (مانند۷۰۹۴) را توانمند ساختند تا برنامه‌ها را بطور دسته ای از نوار خوانده و اجرا کنند.

شکل (۱-۱) یک کامپیوتر بزرگ و گران قیمت ۷۰۹۴ که برای عملیات سنگین محاسباتی استفاده می شده به همراه دو کامپیوتر مجزای ۱۴۰۱ که برای عملیات دریافت اطلاعات اولیه و چاپ نمایش به کار برده می شده نمایش می‌دهد. دقت کنید که ارتباط این سیستم‌ها فقط توسط نوارهای اطلاعاتی[2] است.

---

1 Punched Cards
2 Tapes



شکل ۱-۱: استفاده از دو کامپیوتر ارزان به عنوان ورودی کامپیوتر بزرگ

۱-۲-۳- کامپیوترهای نسل سوم (۱۹۶۵ الی ۱۹۸۰)

با اختراع شدن مدارهای مجتمع[1] نسل سوم کامپیوترها به وجود آمدند. در اوایل دهه ۱۹۶۰ دغدغه شرکتهای بزرگ به وجود آمدن کامپیوترهای مختلف با سخت‌افزارهای مختلف و عدم همخوانی این کامپیوترها با یکدیگر بود. مشتریان این شرکتها انتظار داشتند تا برنامه‌های نوشته شده برای نسخه‌های قبلی کامپیوتها، در نسخه های بعدی قابل استفاده باشد.

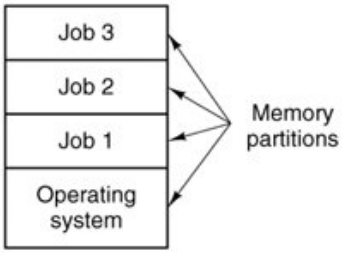
شرکت IBM با معرفی سیستم‌های System/360 این مساله را برای مشتریان خود حل کرد. این سیستم‌ها در قالب یک معماری مشترک برنامه‌پذیر[2] با پیاده سازی های مختلف از کامپیوترهای کوچک معادل ۱۴۰۱ تا کامپیوترهای سنگین تر از ۷۰۹۴ را پوشش می داد و تفاوت در میزان حافظه، سرعت پردازش و میزان ورودی/خروجی این سیستم‌ها بود. کامپیوترهای ۳۶۰ بزرگترین خانواده و معروفترین کامپیوتر های نسل سوم هستند که از تکنولوژی مدارات مجتمع استفاده می کنند و سیستم عامل مخصوصی به نام OS/360 در آنها اجرا می شده است.

مهمترین ویژگی سیستم عامل‌های نسل سوم ، عملکرد چند برنامه‌ای[3] آنها می باشد. پردازنده های کامپیوترهای نسل دوم در حین عملیات ورودی/خروجی مدت زمان طولانی به انتظار می نشستند و بیکار بودند. با به وجود آمدن عملکرد چند برنامه‌ای، کامپیوتر می توانست با تقسیم حافظه به چندین قسمت، چندین برنامه را در آن واحد در حافظه نگهداری کند و

---

1 IC: Integrated Circuits
2 Software-Compatible Machines
3 Multiprogramming

هنگامیکه یک برنامه در انتظار دریافت ورودی یا ارسال خروجی های خود بود، کامپیوتر توانایی آن را داشت تا برنامه بعدی را اجرا کند و از قدرت پردازش پردازنده استفاده بهتری به عمل می‌آمده است.



شکل ۱-۲: نمای یک سیستم چند برنامه‌ای و سه برنامه در حافظه

از دیگر ویژگیهای مهم سیستم عامل های نسل سوم، خواندن کارت‌های پانچ شده و انتقال آنها به دیسک و بار کردن[1] کارهای جدید از روی دیسک، پس از اتمام کارهای در دست اجرا بوده است. این ویژگی به spooling معروف است. به این ترتیب کامپیوترهای واسط ورودی و خروجی ۱۴۰۱ که در نسل دوم استفاده می شدند، بطور کامل از مسیر ورودی/خروجی حذف شدند. تمایل برای پاسخ سریع در کامپیوترها، راه را برای سیستم‌های اشتراک زمانی[2] هموار کرد. ایده کلی این سیستم‌ها در این است که اگر ۲۰ کاربر متصل به کامپیوتر باشند، قدرت کامپیوتر بین این نفرات بطور مساوی بر اساس زمان توزیع می شود و اگر۱۷ نفر آنها در حال گفتگو و خوردن چای باشند، کامپیوتر می تواند سرویس بهتری به ۳ نفر باقی بدهد.

سیستم عامل CTSS اولین سیستم عامل با ایده اشتراک زمان بود که توسط MIT بر روی یک نسخه تغییر یافته کامپیوتر ۷۰۹۴ اجرا می شد. این سیستم عامل تا فراهم شدن بستر مناسب سخت‌افزاری کامپیوترهای نسل سوم عمومی نشد.

پس از موفقیت CTSS گروه MIT، General Electric و Bell Labs تصمیم گرفتند تا پروژه تولید سیستمی را آغاز کنند که توانایی پشتیبانی از چند صد کاربر را بطور همزمان توسط متد اشتراک زمان داشته باشد. این سیستم MULTICS نام داشت و در نوع خود یک

---

[1] Load
2 Timesharing

موفقیت بود. این سیستم عامل با ایده سرویس به چند صد کاربر به وجود آمده و سخت‌افزار آن از نظر پردازش فقط کمی از یک دستگاه Intel 386 فعلی قدرتمندتر بود، ولی تعداد پورت‌های ورودی خروجی آن به مراتب بیشتر بود. اگر چه سخت‌افزار این دستگاه ضعیف به نظر می‌رسد ولی در آن زمان برنامه نویسان به مراتب برنامه های سبک تری می‌نوشتند ، هنری که به مرور زمان از بین رفته است.

پروژه سیستم عامل MULTICS در دنیای کامپیوتر ایده‌ها و راهکارهای منحصر به فردی را به وجود آورد ولی تبدیل پروژه به محصول قابل فروش برای مجریان این پروژه بسیار سخت تر از آن بود که تصور می شد. از مجریان پروژه، Bell Labs و General Electric از پروژه خارج شدند و MIT به تنهایی مسیر را تا موفقیت ادامه داد و توانست تا MULTICS را به یک محصول تبدیل کرده و آن را به فروش برساند. حدود ۸۰ نسخه از این سیستم فروخته شد و همه مشتریان این سیستم که سازمان‌های بزرگ و سرشناس و بسیار باوفا به این سیستم بودند. آخرین ماشین آن در سال ۲۰۰۰ در کانادا خاموش شد.

یکی از متخصصان Bell Labs به نام تامپسون که بر روی پروژه MULTICS کار کرده بود، یک پروژه کوچک شده و تک نفره از MULTICS را شروع کرد که سرانجام به تولید UNIX ختم شد. چون این سیستم عامل در ابتدا متن باز بود[1]، این سیستم عامل به یکی از پر مصرف ترین سیستم عامل‌های دانشگاهی، تجاری و دولتی تبدیل شد و سازمان‌های مختلف شروع به بسط آن در زمینه‌های مختلف و نا هماهنگ با نسخه اولیه کردند.

نسخه‌های معروف تولید شده از UNIX شامل سیستمBSDاز دانشگاه برکلی و System V از AT&T بود. هم اکنون نسخه‌های FreeBSD ، NetBSD و OpenBSD مشتق شده از این نسخه‌های اولیه هستند. برای هماهنگی یونیکس‌ها با یکدیگر و کامپایل شدن و اجرا شدن کد زبان C بر روی یونیکس‌های مختلف، گروه IEEE مجموعه ای از استانداردهای مختلف وضع کرد (POSIX ها) که در ادامه مفصل توضیح داده خواهد شد.

---

1 Open Source Code

۱-۲-۴- کامپیوترهای نسل چهارم (۱۹۸۰ تا امروز)

با به وجود آمدن تکنولوژی LSI (مدارات مجتمع با هزاران ترانزیستور)[1] عمر میکرو پروسسورها (پردازنده های کوچک در مقیاس طولی) شروع شد. مدل‌های مختلفی از کامپیوترهای شخصی که آن روزها میکرو کامپیوتر نامیده می شدند، به وجود آمدند. شرکت اینتل در سال ۱۹۷۴ پردازنده ۸۰۸۰ که اولین پردازنده ۸ بیتی عمومی بود را ارائه کرد. بعضی از شرکتها بر روی این پردازنده یا پردازنده Z80 ، سیستم‌های کاملی را ارائه می دادند که عموما سیستم عاملی با نام CP/M بر روی آنها اجرا می شد.

شرکت موتورولا هم پردازنده ای به نام ۶۸۰۰ را تولید می‌کرد و بعضی مهندسین این شرکت پس از ترک شرکت موتورولا، پردازنده ای به نام ۶۵۰۲ را تولید کردند که در کامپیوتر "اپل ۲" از آن استفاده می شد. چون پردازنده ۶۵۰۲ با ۸۰۸۰ سازگار نبود بعضی از مشتریان اپل ۲ از کارت‌های مخصوصی استفاده می کردند تا بتوانند از امکانات CP/M استفاده کنند. فروشنده این کارت‌ها شرکت کوچکی به نام مایکروسافت بود که بازار مفسر بیسیک[2] را نیز در CP/M در دست داشت.

در اوایل دهه ۱۹۸۰ شرکت اینتل پردازنده ۸۰۸۶ خود را ارائه داد و شرکت IBM سیستمی را بر اساس پردازنده ۸۰۸۸ (مشابه ۸۰۸۶ با ۸ بیت آدرس دهی اضافه) طراحی کرد. در این زمان شرکت مایکروسافت به IBM پیشنهاد ارائه مفسر بیسیک خود به همراه یک سیستم عامل به نام DOS را داد که در اصل تولید شرکت دیگری بود. مایکروسافت DOS را خرید و مولف اولیه را به کار گرفت تا آن را بهینه کند. محصول تولید شده MS-DOS نام گرفت.

سیستم‌های عامل CP/M ، MS-DOS و Apple DOS همگی سیستم عامل هایی با خط فرمان[3] بودند و در کل ظاهر گرافیکی نداشتند. در اوایل سال ۱۹۸۴ شرکت اپل مکینتاش اولین سیستم‌های گرافیکی را مبتنی بر پردازنده ۶۸۰۰۰ که پردازنده ۱۶ بیتی شرکت موتورولا است، ارائه داد.

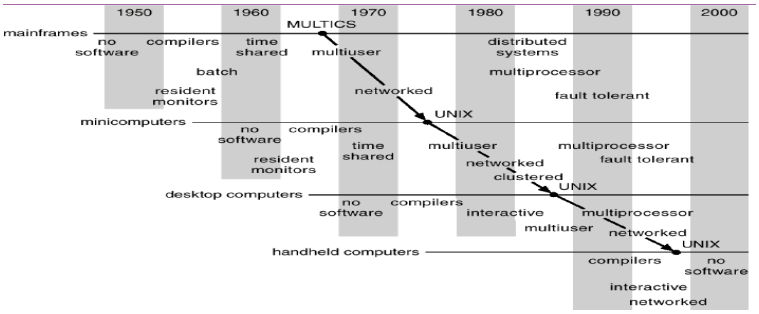
---

1 LSI: Large Scale Integration
2 Basic Interpreter
3 Command-line Systems

برای مقابله با شرکت اپل، شرکت مایکروسافت نسخه‌های اولیه ویندوز خود را ارائه داد که در اصل فقط یک لایه بر روی سیستم عامل DOS خود بود. امروزه نرم‌افزار ویندوز با هسته سیستم‌عامل Windows NT ارائه می‌شود که یک سیستم عامل ۳۲ بیتی است و مایکروسافت آن را مجددا از ابتدا نگارش کرده است.

یک سیستم عامل معروف دیگر در دنیای کامپیوترهای شخصی، یونیکس است که معمولا در کامپیوترهای با پردازنده‌های RISC استفاده می‌شود. در کامپیوترها خانواده اینتل و متشابهات آن مانند AMD، معمولا از سیستم عامل لینوکس استفاده می‌کنند که سیستم عامل مناسبی در کاربردهای عمومی و دانشگاهی است و روز به روز بر مصرف و کاربران آن اضافه می شود.

شکل (۱-۳) نشان دهنده رشد سیستم عامل‌ها را از سال ۱۹۵۰ تا سال ۲۰۰۰ می باشد.



شکل ۱- ۳: تاریخچه سیستم عامل‌ها در یک نگاه

در اواسط دهه ۱۹۸۰ و با گسترش شبکه کامپیوترهای شخصی، سیستم عامل‌های مبتنی بر کامپیوترهای شخصی متصل به شبکه Network Operating Systems (سیستم عامل‌های شبکه) و Distributed Operating Systems (سیستم عامل‌های توزیع شده) هم جایگاه های خاص خود را پیدا کردند.

سیستم عامل‌های شبکه فرق چندانی با سیستم عامل‌های معمولی ندارند و اختلاف در این است که از طریق شبکه کاربران می توانند به آنها وارد شوند و از آنها استفاده کنند؛ برای مثال فایل‌های خود را کپی کنند. معمولا در سیستم عامل‌های شبکه هر سیستم کاملا مستقل از سیستم دیگر است.

سیستم عامل‌های توزیع شده معماری داخلی مشابهی با سیستم عامل‌های معمولی دارند، با این تفاوت که برنامه‌ها در حین اجرا بر روی چندین کامپیوتر اجرا می شوند و از منابع چندین

کامپیوتر استفاده می کنند. در سیستم عامل‌های توزیع شده کاربران و برنامه‌ها دقیقا نمی‌دانند که منابع سیستم مثل فایل‌ها در کدام کامپیوتر یا کامپیوترها به صورت فیزیکی ذخیره می‌شوند.

۱-۳- انواع سیستم عامل ها

می‌توان سیستم عامل‌ها را بسته به مقیاس اندازه، کارایی و سخت‌افزار کامپیوتر به موارد زیر تقسیم کرد که در زیر فهرست شده‌اند:

۱-۳-۱-سیستم عامل های Mainframe

این سیستم عامل‌ها ویژه کامپیوترهای مادر[1] هستند. کامپیوترهای مادر در مراکز اطلاعات و سازمان‌های بزرگ یافت می‌شوند و صد البته سیستم عامل‌های این کامپیوترها با چندین پردازش‌گر، صدها دیسک سخت و چندین گیگابیت حافظه با سیستم عامل‌های کامپیوترهای شخصی تفاوت‌های بسیاری دارند.

این سیستم عامل‌ها به سخت‌افزار خود کاملا وابستگی دارند و معمولا برای اجرای چندین کار در آن واحد طراحی و بهینه شده اند. معمولا این سیستم عامل‌ها یکی از سه نوع سرویس Batch،Transactionو Time sharingرا ارائه می‌کنند. سرویس Batch برای اجراء کارها بطور پشت سرهم و دسته‌ای می باشد و محاوره چندانی با کاربر ندارد. مثال این سرویس محاسبه مالی یا آماری بسیار بزرگ است. سرویس Transaction برای اجرای هزاران عملیات کوچک و سبک در آن واحد است. مثال این سرویس عملیات بانکی مانند حساب‌ها و چک‌ها است که معمولا زمان زیادی نمی‌گیرد و سیستم بهینه می شود تا بتواند صدها درخواست را در آن واحد جوابگو باشد. سرویس Time sharing برای اجرای همروند چندین عملیات در یک زمان است، مانند اتصال چندین کاربر به یک بانک اطلاعاتی بزرگ و استخراج اطلاعات از این بانک. این سیستم مشابه سیستم عامل های کامپیوترهای شخصی امروزی است.

---

[1] Mainframe

۱-۳-۲- سیستم عامل های سرور

این سیستم عامل‌ها معمولا کوچک شده‌ی سیستم عامل‌های کامپیوترهای مادر هستند و معمولا بر روی کامپیوترهای شخصی بزرگ، کامپیوترهای مادر و متشابهات آنها که برای همین کار طراحی شده اند اجرا می شوند. این سیستم عامل‌ها وظیفه دارند منابع سیستم مانند فایل‌ها و پردازنده را بین کاربران متعدد به اشتراک قرار دهند. اینترنت بر روی چنین سیستم عامل‌هایی رشد کرده و می‌کند. مثال این سیستم عامل‌ها خانواده UNIX ، Windows 2000 و Linux است.

۱-۳-۳- سیستم عامل های چند پردازنده ای

عموماً یکی از روش‌های ارتقاء قدرت سیستم، قرار دادن چند پردازنده در درون یک کامپیوتر است. این سیستم‌ها بسته به طراحی نحوه قرارگیری و ارتباط این پردازنده‌ها و بسته به منابع به اشتراک قرار داده شده، نامهای مختلف دارند و آنها را کامپیوترهای موازی[1]، کامپیوتر های چند گانه[2] و کامپیوترهای چند پردازنده[3] می نامند. سیستم عامل‌های این نوع کامپیوترها تفاوت‌های مشهودی با سیستم‌های دیگر دارند و معمولا با توجه به سرویسی که ارائه می‌دهند در خانواده سیستم‌عامل‌های سرور قرار می گیرند. در ادامه این سیستم‌ها بیشتر بررسی شده اند.

۱-۳-۴- سیستم عامل های کامپیوترهای شخصی

وظیفه اصلی این سیستم عامل‌ها ارائه محیط و امکانات مناسب به تنها کاربر سیستم است. این سیستم‌عامل‌ها به طور عمده برای عملیاتی مثل واژه پردازی[4] (Word Processing) ، صفحات گسترده[5] و غیره استفاده می شوند. سیستم‌عامل ویندوز XP/98 و سیستم‌عامل مکینتاش و لینوکس نمونه های بارز این گونه سیستم‌ها هستند.

---

1 Parallel Computers
2 Multi Computers
3 Multi Processors
4 Word Processing
5 Spread sheets

۱-۳-۵-سیستم عامل های بلادرنگ

در این سیستم‌عامل‌ها، زمان به عنوان یک پارامتر اصلی مطرح می باشد. از این سیستم عامل‌ها بخصوص در صنعت برای کنترل به موقع ماشین‌ها استفاده می کنند. برای مثال در خط مونتاژ در حال حرکت ماشین‌ها، روبات‌ها موظف هستند در زمان خاصی، عملکرد خاص خود را انجام دهند. اگر یک روبات کمی زودتر یا دیرتر از زمان لازم عمل کند، ممکن است محصول خط تولید را کلا از بین ببرد یا خسارات جبران ناپذیر به محصول وارد کند .

سیستم‌های بلادرنگ معمولا به دو نوع اصلی سیستم‌عامل‌های سخت و سیستم عامل‌های نرم تقسیم می‌شوند. سیستم عامل‌های بلادرنگ سخت، انجام شدن یک عملیات را در زمان خود تضمین می‌کنند. در این سیستم‌ها معمولا بسیاری از ویژگی‌های پایه ای سیستم عامل‌ها مانند مدیریت حافظه مجازی، اشتراک زمانی و چند کاربری به علت اتلاف زمان در عملکرد سیستم، حذف می‌شوند و هدف اولیه این سیستم عامل‌ها فقط اجرای به موقع برنامه است.

سیستم عامل‌های بلادرنگ نرم بسیار مشابه سیستم عامل‌های معمولی هستند با این تفاوت که یک برنامه نسبت به سایر برنامه‌ها اولویت اجرای بیشتری دارد و تا اتمام اجرا، برنامه‌ی دیگری اجرا نخواهد شد. این سیستم عامل‌ها نمی توانند در مناطق بسیار بحرانی مانند روبات‌ها و سیستم‌های ناوبری هواپیماها، استفاده شوند ولی در بسیاری کاربردهای دیگر مانند سیستم‌های چند رسانه ای[1] استفاده می شوند.

۱-۳-۶- سیستم عامل های تعبیه شده

این سیستم عامل‌ها در لوازم الکتریکی مانند موبایل، تلویزیون، میکرو ویو، در اتومبیل‌ها، در صنعت و غیره استفاده می شوند. این سیستم عامل‌ها معمولا فقط عملیات خاص و تعریف شده‌ای را انجام می دهند و از سخت‌افزارهای محدودی پشتیبانی می‌کنند. این سیستم عامل‌ها معمولا چندان در دید کاربر نیستند. مثال بارز این سیستم عامل‌ها Windows CE ، Palm OS و Symbian است.

---

1 Multimedia

سیستم عامل‌های تعبیه شده چون رابط کاربری زیادی با کاربر ندارند و کاربران آنها معمولا غیر متخصص هستند مسائلی چون پایداری سیستم عامل در مقابل خطاهای کاربری و برنامه‌های کاربردی موجود در آنها بسیار اهمیت دارند. معمولا در این سیستم‌ها مسائلی چون لایه‌های امنیت خارجی و داخلی و چند کاربری نیز بنا به کاربرد تعبیه نمی شوند یا بسیار محدود است.

۱–۳–۷–سیستم عامل های کارت های هوشمند

کوچکترین نوع سیستم عامل، سیستم عامل‌های موجود در کارت‌های هوشمند است. این کارت‌ها که معمولا شامل یک پردازنده و مقدار محدودی از حافظه هستند، برای عملیاتی بسیار محدود تعبیه شده‌اند. بعضی از این سیستم عامل‌ها فقط یک روتین مانند دریافت/پرداخت پول انجام می دهند در حالیکه بعضی دیگر چندین عملیات انجام می دهند.

بعضی از سیستم عامل های کارت‌های هوشمند امکانات ماشین مجازی جاوا را ارائه می‌دهند، به این معنی که می توانند یک برنامه کوچک نوشته شده به زبان جاوا را اجرا کنند.

در سیستم عامل‌های کارت‌های هوشمند محدودیت‌های سخت‌افزار منجر می‌شود تا سیستم عامل‌های بسیار سبک و کوچک تولید شود که گاهی به فقط به زبان اسمبلی برای یک نوع مدل یا دستگاه نگارش می شوند و اکثرا به سخت‌افزار خود وابسته بوده و با دستگاه توسط تولید کننده ارائه می شوند.

۱–۴– انواع ساختار های داخلی سیستم عامل (ساختارهسته)

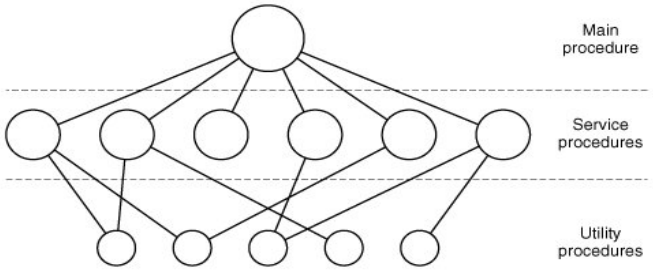
در اینجا پنج ساختار معروف هسته سیستم‌عامل شامل؛ ۱- سیستم های یکپارچه[1] ۲-سیستم‌های لایه‌ای[2] ۳- سیستم‌های ماشین مجازی[3] ۴- سیستم های Exokernels ۵- سیستم های سرویس دهنده و سرویس گیرنده[4] می پردازیم.

[1] Monolithic
[2] Layered
[3] Virtual Machine
[4] Client-Server Model

۱-۴-۱- سیستم های یک پارچه

معمولترین سیستم‌ها تا کنون سیستم‌های یک پارچه بوده اند. در این سیستم‌ها هیچ ساختاری خاصی وجود ندارد و سیستم عامل از چندین و چند تابع تشکیل شده که همگی در یک سطح قرار دارند و می توانند یکدیگر را بدون هیچ محدودیت فراخوانی کنند. این سیستم‌ها معمولا در یک فایل دودویی (ماشینی) قابل اجرا برای کامپیوتر هستند که کل سیستم عامل در آن تعبیه شده است. نحوه ارتباط این سیستم عامل‌ها با برنامه های کاربردی به صورت تعبیه کردن محل‌های خاص برای قرار دادن اطلاعات توسط برنامه‌ها (مانند رجیسترها یا پشته) و اجرای کد مخصوص مربوط به Kernel Call یا Supervisor Call (مانند یک وقفه نرم‌افزاری) است. این کد خاص کنترل کامپیوتر را به سیستم عامل می دهد و سپس مد پروسسور از مد کاربری به مد سیستمی منتقل می شود و برنامه‌ها و دستورالعمل‌های سیستمی در این مد اجرا شده و کنترل مجددا به برنامه کاربر منتقل می شود.

توجه کنید که بسیاری از پروسسورها دارای دو مد کاری هستند. در مد سیستمی تمامی دستورات ماشین قابل اجرا است در حالیکه در مد کاربری معمولا دستورات ورودی خروجی و بسیاری دستورات پر اهمیت غیر قابل اجرا است.



شکل ۱-۴: یک سیستم یک پارچه

۱-۴-۲-سیستم های لایه ای

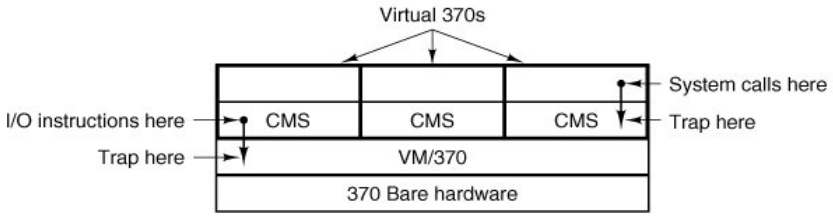
همانطور که از شمایل سیستم‌های یک پارچه بدست می آید، راه دیگر پیاده سازی سیستم عامل تعبیه لایه‌های داخلی برای سیستم عامل است. در سیستم‌های لایه‌ای هر لایه وظیفه خاصی دارد و هر لایه فقط با لایه بالاتر و پایین تر از خود ارتباط دارد. برای مثال در سیستم

عامل قدیمی هلندی THE که از ۶ لایه تشکیل شده، لایه صفرم برای تخصیص و اشتراک پروسسور و لایه یکم برای مدیریت حافظه استفاده می شده است.

سیستم عامل معروف MULTICS از تکنولوژی مشابه این تکنولوژی استفاده می کرده است.

۱-۴-۳- سیستم های ماشین مجازی

ایده سیستم‌های مجازی از کامپیوترهای مادر OS/ 360 شرکت IBM شروع شده است. این کامپیوترها در ابتدا به صورت یک پارچه مدیریت می شده اند و برنامه‌ها در یک فضا اجرا می‌شده اند، از این رو اجرای چندین برنامه بطور همزمان با متدهای اشتراک زمانی و غیره وجود نداشته است. یک تیم توسعه در شرکت IBM سیستم عامل VM/ ۳۷۰ را تولید کرد. این سیستم عامل کامپیوتر مادر ۳۷۰ را به چندین کامپیوتر مجازی با مشخصات مطابق با کامپیوتر مادر تبدیل می کرد. به این صورت در یک ماشین چندین برنامه مخصوص۳۷۰ می‌توانستند اجرا شوند و هر کدام تصور کنند که کل ماشین مادر را در اختیار دارند. این سیستم عامل بر خلاف سیستم‌های عامل معمول، مدل برنامه سازی سخت‌افزار کامپیوتر را تعمیم نمی دهد و فقط امکان اشتراک منابع مانند پردازنده و غیره را به وجود می آورد و مجددا سخت‌افزار شبیه سازی شده را به برنامه های خود ارائه می کند. در حال حاضر هنوز از این سیستم عامل بطور گسترده استفاده می شود.



شکل ۱-۵: سیستم‌عامل VM/ ۳۷۰

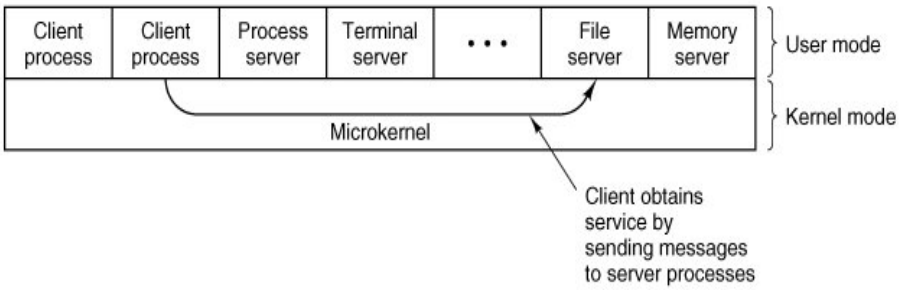
۱-۴-۴-سیستم های Exokernels

این سیستم‌ها مانند سیستم‌های ماشین‌های مجازی، کامپیوتر را به چندین کامپیوتر تقسیم می کنند. با این تفاوت که در ایده ماشین‌های مجازی هر برنامه تصور می کند که کل ماشین در اختیار او است ولی در Exokernels هر برنامه می داند که فقط قسمتی از یک کامپیوتر

بزرگ در اختیار او است. این ویژگی سبب می شود تا مسئولیت ترجمه آدرس‌ها و منابع در Exokernels از دوش سیستم عامل برداشته شود. وظیفه اصلی Exokernels مراقبت از آدرس دهی ماشین‌های مجازی داخلی خود و جلوگیری از تجاوز هر ماشین به فضای ماشین دیگر (از طریق درخواست آدرس‌های اشتباه) و تخصیص منابع به آنهاست.

۱-۴-۵- سیستم های سرویس دهنده و سرویس گیرنده

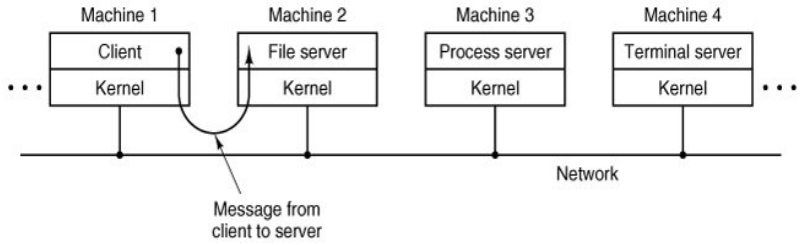
ایده کلی در سیستم‌های سرویس دهنده و سرویس‌گیرنده کوچک نگه داشتن فضای کد مربوط به هسته مرکزی و انتقال عملیات اجرایی به فضای کاربری است. در این نوع سیستم عامل‌ها، هسته مرکزی تنها ارتباط برنامه‌های کاربردی با برنامه‌های خادم (سرور) فایل و حافظه را بر قرار می کند.



شکل ۱-۶: سیستم‌های سرویس دهنده و سرویس گیرنده

یکی از مزایای این سیستم‌ها این است که چون هیچ یک از خادم‌ها دسترسی مستقیم به سخت‌افزار ندارند و در فضای کاربران و با مد کاربری اجرا می شوند، در صورت بروز هر مشکلی فقط خادم مربوطه از کار می افتد و ماشین اصلی دچار مشکل نمی شود.

از دیگر مزایای این سیستم امکان استفاده آن در سیستم عامل‌های توزیع شده است. در صورتیکه یک برنامه یکی از منابع سیستم را تقاضای کند، این تقاضا می تواند بر روش شبکه منتقل شده و توسط یک خادم در یک کامپیوتر دیگر پاسخ داده شود.



شکل ۱-۷: سیستم عامل‌های سرویس دهنده و سرویس گیرنده (توزیع شده)

۱-۵- فراخوانی سیستمی و مفاهیم سیستم عامل

برای شناخت سیستم عامل و مفاهیم آن، می‌باید نحوه ارتباط سیستم عامل با برنامه های کاربردی را نیز بررسی کرد. ارتباط سیستم عامل با برنامه‌های کاربردی به صورت یک سری دستور العمل واسط است که سابقاً این ارتباط به نام فراخوانی سیستمی[1]معروف بود. برای درک صحیح سیستم عامل، شناخت این ارتباط و نحوه آن امری ضروری است.

اگرچه این ارتباط در سیستم عامل‌های مختلف، یکسان نیست ولی مفاهیم کلی آن در همه سیستم عامل‌ها مشترک است. در زمینه ارتباطات سیستم عامل با برنامه‌های کاربردی، استانداردهایی از قبیل استاندارد POSIX وجود دارد که سعی در استاندارد کردن و یکسان کردن ارتباط سیستم عامل با برنامه‌های کاربردی دارند. بسیاری از سیستم عامل‌های معروف مانند یونیکس، لینوکس، خانواده BSD و حتی ویندوز (تا حدودی) از این استاندارد پشتیبانی می کنند.

یکسان شدن این ارتباط و تبعیت سیستم عامل‌ها از این استانداردها به برنامه نویس این امکان را می دهد که برنامه خود را در یک سیستم عامل واجد شرایط این استاندارد نوشته و برنامه خود را به سادگی و بدون دغدغه در لایه کد منبع به بقیه سیستم عامل‌های تابع این استاندارد انتقال دهد. در ادامه به توضیح بیشتر استانداردهای معروف می پردازیم.

---

[1] System Calls

۱-۵-۱- استانداردهایIEEE، ساختار واسط سیستم عامل های قابل حمل[1]POSIX

در این استاندارد، واسط یک سیستم عامل استاندارد و محیط آن[2] (که همگی به زبان C هستند)، به همراه یک پوسته[3] و تعدادی برنامه کاربردی که کمک می کند تا برنامه‌ها در لایه کد منبع[4] قابل انتقال باشند، تعریف و استاندارد شده است. این استانداردها از یک پروژه در حدود سال ۱۹۸۵ پدیدار شدند و همچنان در حال تکامل هستند. درحال حاضر (بهار ۱۳۸۵) ، نسخه سال ۲۰۰۴ این استانداردها مطرح است که IEEE و Open Group به صورت مشترک در کمیته ای به نام PASC[5] به همراه سایر اعضا و علاقمندان در این زمینه، این استانداردها را تنظیم می کنند. نام POSIX را ریچارد استالمن در درخواست IEEE برای نامی که در اذهان می ماند، پیشنهاد کرده است.

در حال حاضر این استاندارد در حدوداً ۱۵ سند مختلف تعریف شده است و به لحاظ موضوعی به سه بخش عمده زیر تقسیم می شود:

- توابع مربوط به هسته سیستم عامل که در آن استاندارد POSIX.1 ، سرویس های بلادرنگ و توسعه های آن، واسط چند ریسمانی، واسط امنیتی و شبکه‌ها، هم وجود دارند.
- دستورات و برنامه های کاربردی
- آزمونهای بررسی تطبیق با این استاندارد

سیستم عامل‌های لینوکس، خانواده BSD و حتی بسیاری از سیستم‌های آکادمیک مانند MINIX (تا حدودی) از این استاندارد پشتیبانی می کنند و این سبب می شود تا بتوان کد برنامه‌های زبان C را به این سیستم‌ها انتقال داد.

---

1 Standard for Information Technology - Portable Operating System Interface (POSIX)
2 Standard operating System Interface and Environment,
3 Shell
4 Source Code
5 Www.pasc.org

۱-۵-۲- استانداردهای Open Group، تنها توصیف رسمی یونیکس

این استانداردها که به موازات استانداردهای POSIX و با هدف استاندارد کردن یونیکس تکمیل می شده اند، توسط گروه Open Group تنظیم می شده و در حال حاضر باIEEE در گروه PASC بطور مشترک فعالیت می کنند. گروه Open Group مالک اصلی نام UNIX است و نام استانداردهای خود را باSingle Unix Specification می نامد. اعضای این گروه شرکت‌هایی مانند IBM ، HP ، NEC و شرکت‌های معروف دیگر هستند.

۱-۵-۳- استانداردهای پایه لینوکس[1]

در کنار استانداردهای معروفی چون POSIX و Single Unix Specification استانداردی با نام LSB یا استانداردهای پایه ای لینوکس وجود دارد که هدف آن استاندارد کردن توزیع‌کنندگان مختلف لینوکس است. این استاندارد به علت تنوع توزیع‌های لینوکس، برای استاندارد کردن محیط و کتابخانه‌های یک توزیع کننده لینوکس است و توسط گروهی به نام گروه استانداردهای رایگان[2] تهیه می شوند. اعضای این گروه شرکت‌هایی مانند IBM ، Red Hat، Novel، Sun Microsystems، Google ، Debian و بسیاری از شرکت‌های معتبر دیگر هستند.

توزیع‌های لینوکس که از این استاندارد حمایت می کنند، دارای محیط و کتابخانه‌های مشترک هستند و انتقال برنامه‌ها به این توزیع‌کنندگان یکسان است. این استاندارد، استانداردهای POSIX و Single Unix Specification را در دل خود جا داده است و علاوه بر آن کتابخانه‌هایی مانند X Window و فایل سیستم‌ها را می پوشاند.

محصولاتی چون Red Hat Enterprise Linux ، SUSE Linux Enterprise Sever و Asia Linux مطابق این استانداردها هستند و از این استانداردها پشتیبانی می کنند در حالیکه توزیع‌های معروفی مانند Debian و Slackware به علت ماهیت و بایدها و نبایدها در این استاندارد، آنچنان از این استانداردها استقبال نکرده اند یا در قسمت‌هایی مغایرت دارند.

---

1 LSB : Linux Base Standards
2 http://freestandards.org , Free Standard Group

۱–۶– سیستم عامل های کامپیوترهای دارای چند پردازنده[1]،سیستم های موازی[2]

از آغاز به وجود آمدن صنعت کامپیوتر، تلاش برای تقویت این دستگاه هرگز خاتمه نیافته است و همه روزه تقاضای برای کامپیوترهای قوی تر به وجود می آید. هنگامی که دستگاه ENIAC[3] (اولین و بزرگترین کامپیوتر برنامه پذیر که توسط ارتش آمریکا در سال ۱۹۴۲ تولید شده است) تولید شد، هزار برابر سریع تر از تمام محاسبه گرهای زمان خود بود و می‌توانست سیصد دستورالعمل را در هر ثانیه اجرا کند. کامپیوترهای امروزی بیش از یک میلیون بار از آن دستگاه قوی تر هستند ولی هنوز تلاش برای تقویت کامپیوترها خاتمه نیافته است.

در قدیم راه کار قابل قبول، افزایش سرعت فرکانس کاری پردازنده بود، ولی متاسفانه این صنعت هم اکنون به محدودیت های فیزیکی عناصر رسیده است و ایده افزایش فرکانس دیگر را چاره نیست. بنا به نظریات هیچ سیگنالی سریع تر از سرعت نور نمی تواند حرکت کند که ۳۰ سانتی متر بر نانو ثانیه در خلاءو ۲۰ سانتی متر بر نانو ثانیه در فلز مس یا فیبر نوری است. این به این مفهوم است که با یک کامپیوتر فرضی با سرعت ساعت (کلاک) ۱۰ گیگا هرتز ، سیگنال نمی تواند بیش از ۲ سانتی متر حرکت کند و با یک کامپیوتر با سرعت ۱۰۰ گیگا هرتز کلاک کمتر از ۲ میلی متر حرکت می کند. با یک سرعت ۱ ترا هرتز این مسافت به ۱۰۰ میکرون کاهش می یابد.

شاید ساختن کامپیوترهایی با این ابعاد ممکن باشد ولی نمی توان مشکل حاصل از تولید گرما را در دستگاه‌های الکترونیکی و پردازنده‌ها را به این سادگی در این ابعاد حل کرد. هرچه کامپیوتر سریع تر کار کند، گرمای تولیدی حاصل افزایش می یابد و هرچه ابعاد کوچکتر باشد سخت تر می توان با گرمای تولید شده مقابله کرد و آن را دفع کرد. هم اکنون سیستم خنک کننده پردازنده‌های پنتیوم از خود پردازنده بزرگتر است و با افزایش سرعت بیشتر این سیستم‌ها، به سیستم‌های خنک کننده بزرگتر و بهتر احتیاج دارند.

---

1 Multiple Processor Systems
2 Parallel Systems
3 Electronic Numerical Integrator and Computer

یک راه حل افزایش سرعت و قدرت کامپیوترها، موازی ساختن کامپیوترها با یکدیگر است. با موازی ساختن و جمع کردن پردازنده‌ها در کنار یکدیگر، سرعت چندین برابر خواهد شد. هم اکنون سیستم‌هایی با بیش از ۱۰۰۰ پردازنده در بازار موجودند.

کامپیوترهای موازی برای محاسبات سنگین استفاده می شوند، محاسباتی نظیر پیش بینی هوا در سازمان هواشناسی و محاسبات مهندسی و شبیه سازی‌های مختلف مانند بال یک هواپیما. انجام این گونه عملیات ریاضی به زمانی بسیار طولانی احتیاج دارند که این عملیات را می توان در چندین پردازنده بطور همزمان تقسیم کرد.

از آنجا که پیاده سازی های سخت‌افزاری کامپیوترهای دارای چند پردازنده مختلف است، پیاده سازی سیستم عامل‌های چند پردازنده هم با یکدیگر تفاوت دارند. یکی از معمولترین پیاده سازی‌ها SMP[1] (چند پردازنده‌ای متقارن) نام دارد. در این پیاده سازی یک نسخه سیستم عامل در حافظه اصلی وجود دارد و هر پردازنده‌ای می تواند آن را اجرا کند. این سیستم با تعبیه قفل‌های نرم‌افزاری در فضای سیستم عامل مانع از تداخل عملکرد پردازنده‌ها با یکدیگر شده و سبب می شود تا در آن واحد فقط یک پردازنده کد سیستم عامل یا بخش‌های خاصی از سیستم عامل که با یکدیگر تداخل دارند را اجرا کند و بقیه یا منتظر می‌مانند و یا مشغول اجرای برنامه‌های کاربر هستند. در این معماری ممکن است با افزایش تعداد پردازنده‌ها، زمان بیکاری و اتلاف وقت در پردازنده‌ها با توجه به نوع عملکرد سیستم افزایش یابد.

در یک مدل پیاده سازی دیگر که به چند پردازنده ای نامتقارن[2] معروف است، هر پردازنده به وظیفه معینی می پردازد و یک پردازنده ارشد وجود دارد که به عملکرد همه پردازنده‌ها نظارت می کند. دراین مدل به سادگی می توان به هر پردازنده یک پردازش را منسوب کرد و راندمان کار بسیار بالایی داشت.

---

1 Symmetric Multi Processing Model
2 Asymmetric Multi Processing

۱-۷- سیستم های توزیع شده[1] و سیستم عامل های توزیع شده[2]

کامپیوترهای چند پردازنده و کامپیوترهای بزرگ مانند Mainframeها قدرت پردازش فراوانی نسبت به سیستم‌های معمول دارند ولی مساله اصلی در قیمت این سیستم‌ها است. با به وجود آمد تکنولوژی‌هایی مانند شبکه های محلی پر سرعت، سرعت ارتباطات کامپیوترها بسیار افزایش یافته است و از این رو می توان با استفاده از چندین کامپیوتر متصل به یکدیگر، درآن واحد از مجموعه ای از کامپیوترها برای پردازش یک عملیات بزرگ استفاده کرد.

سیستم‌های توزیع شده پیاده سازی‌ها و تعاریف مختلفی دارند، ولی معمولترین تعریف برای آنها عبارت است از دو یا چند کامپیوتر مستقل و متصل به یکدیگر که از دید کاربر یک کامپیوتر دیده می شوند و منابع خود مانند قدرت پردازش و حافظه را با یکدیگر به اشتراک می‌گذارند.

تعریف فوق دو محور اصلی مستقل بودن سیستم‌ها از یکدیگر و یکپارچه بودن سیستم در نگاه کاربر را در خود جاداده است. برای مثال در یک دانشگاه یا شرکت، شبکه کامپیوترهای متصل به هم که دارای سیستم توزیع شده مخصوص هستند می توانند در آن واحد هم به کاربر خود سرویس بدهند و هم می توانند بار کلی سیستم که بر روی آنها توزیع شده را به دوش بکشند به این صورت یک کاربر ممکن است برنامه‌ای با زمان پردازش طولانی به اجرا بگذارد و این برنامه از مجموعه توان همه کامپیوترها نهایت استفاده را ببرد. براساس اخبار اینترنتی، شرکت گوگل در ابتدای عمر خود از این معماری استفاده می‌کرده است، به این صورت که تمام کامپیوترهای پرسنلی این شرکت در زمان بیکاری خود قدرت سرویس دادن به مشتریان این شبکه و پردازش جستجوهای مورد نظر مشتریان اینترنتی را دارابودند. هم‌اکنون هم گوگل از مشتریان خود می خواهد تا با عضو شدن در پروژه آزمایشگاه اینترنتی گوگل[3]، به این شرکت امکان استفاده از قدرت پردازش کامپیوتر خود برای کارهای بین المللی تحقیقاتی که جنبه‌های عمومی و پزشکی دارند، بدهند.

---

1 Distributed Systems
2 Distributed Operating Systems
3 Google Labs

قبل از بررسی سیستم عامل‌های توزیع شده، سیستم‌های توزیع شده (مانند برنامه های توزیع شده) را بررسی می کنیم تا ایده کلی سیستم عامل های توزیع شده ملموس تر شود و در نهایت به سیستم عامل های توزیع شده می پردازیم. در ادامه مزایا و معایب سیستم‌های توزیع شده به اختصار بیان شده است.

۱–۷–۱– مزایای سیستم های توزیع شده نسبت به سیستم های متمرکز

اولین و بزرگترین مزیت سیستم‌های توزیع شده نسبت به سیستم‌های متمرکز، قیمت تمام شده این سیستم‌ها ست. این سیستم‌ها به سخت‌افزار ارزان تری برای اجرا نیازدارند.

مزیت دیگر سرعت این سیستم‌ها است، در حدود یک چهارم قرن قبل، طبق یک قانون که به نام مولف آن گروش[1] معروف است، بیان شده که نسبت بهایی که به کامپیوتر می دهید با دو برابر قدرت آن کامپیوتر متناظر است؛ به این معنی که اگر برای یک کامپیوتر دو برابر پول بیشتر نسبت به یک کامپیوتر معمول دیگر هزینه کنید، چهار برابر کامپیوتر بهتری می خرید. این قانون دقیقا برای Mainframeها در زمان خود درست عمل می‌کرده است و سبب می شده تا هر کسی نهایت پول مقدور خود را برای بهترین کامپیوتر مورد نیاز خود بپردازد. امروزه با وجود ریز پردازنده‌ها و کامپیوترهای شخصی، این قانون صحت خود را از دست داده است و عموما اگر شما بخواهید برای یک کامپیوتر مناسب یک میلیون تومان بپردازید،نمی توانید با پرداخت دو میلیون تومان یک کامپیوتر با امکانات چهار برابر قوی تر تهیه کنید به این ترتیب باخرید مجموعه ای از کامپیوترهای ارزان قیمت و استفاده از برنامه های توزیع شده، می توانید به قدرت پردازش بالاتری برسید.

مزیت سوم سیستم‌های توزیع شده، قابلیت اطمینان این سیستم‌ها است. در سیستم‌های توزیع شده، برای مثال یکصد کامپیوتر، از بین رفتن پنج درصد سیستم یا دراین مثال، پنج کامپیوتر صدمه آنچنانی به سیستم اصلی وارد نمی کند و در یک پیاده سازی خوب سیستم به سادگی به حیات خود ادامه می دهد. در مناطقی که امنیت پارامتر بسیار مهمی محسوب می شود، مانند هواپیماها، سیستم‌های توزیع شده نقش پررنگ تری دارند.

---

1 Grosch

مزیت چهارم، قدرت افزایش تدریجی این سیستم‌ها ست. در راهکارهای متمرکز مانند Mainframeها، در صورت افزایش بار دستگاه، یا باید دستگاه را با یک مدل بزرگتر عوض کرد و یا یک دستگاه دیگر خرید که هر دو به لحاظ هزینه، بار سنگینی به دوش مجموعه قرارمی‌دهند؛ در صورتیکه در سیستم‌های توزیع شده فقط چند کامپیوتر ساده به مجموعه اضافه می شود.

۱-۷-۲- مزایای سیستم های توزیع شده نسبت به کامپیوترهای مستقل

کامپیوترهای مستقل ساده ترین مدل کاری محسوب می شود. در این مدل هر کامپیوتری خود به تنهایی بار خود را به دوش می کشد و تنها قسمتی از منابع خود را توسط شبکه محلی مانند چاپگرها و فایل‌ها را با کامپیوترهای دیگر به اشتراک قرار می دهند.

اگرچه مدل کامپیوترهای موجود در یک شبکه محلی که اطلاعات و منابع خود را برای یکدیگر به اشتراک قرار می دهند و از یک یا چند سرور اصلی استفاده می کنند، بسیار نزدیک به مدل سیستم‌های توزیع شده است ولی چون در این مدل، از منابع بسیار مهم مانند قدرت پردازش کامپیوترها استفاده ای نشده است، این سیستم‌ها توزیع شده محسوب نمی شوند. در این سیستم‌ها همچنان بار هر کامپیوتر و بخصوص سرورها بردوش خود دستگاه است، در حالیکه کامپیوترهای بسیاری از پرسنل فقط روشن هستند و خاک می خورند.

۱-۷-۳-معایب سیستم های توزیع شده

گرچه سیستم‌های توزیع شده مزایای زیادی دارند ولی معایبی نیز دارند. بزرگترین عیب در سیستم‌های توزیع شده، نرم‌افزارهای توزیع شده است. سیستم عامل‌های توزیع شده، زبان‌های برنامه سازی توزیع شده،برنامه‌های کاربردی توزیع شده و... همگی ناشناخته تر و کمیاب تر از نسخه های توزیع نشده هستند.

نقطه ضعف دیگر سیستم‌های توزیع شده، شبکه اتصال این کامپیوترها به یکدیگر هستند، چون این سیستم‌ها با ارسال پیام با یکدیگر ارتباط برقرار می کنند، از بین رفتن پیام‌ها در اتصالات، تخریب به بار خواهد آورد و سیستم های توزیع شده احتیاج به سیستم‌های کنترلی و بازیابی شدید دارند که وجود این سیستم‌ها بار کلی سیستم را افزایش می دهند.

نقطه ضعف آخر این سیستم‌ها، امنیت آنهاست. چون در این سیستم‌ها منابع مانند فایل‌ها در کل کامپیوترها پراکنده می شوند، برقراری امنیت منابع، یکی از پارامترهای بسیار مهم در این سیستم‌ها خواهد بود.

با وجود این معایب به نظر می رسد سیستم‌های توزیع شده در سال‌های آینده، رشد بسیاری خواهند داشت و بسیاری از شرکت‌ها وسازمان‌ها برای استفاده صحیح از منابع خود و پایین آوردن هزینه‌ها، به سیستم‌های توزیع شده روی خواهند آورد.

۱-۷-۴- سیستم عامل های توزیع شده

پس از بررسی سیستم‌های توزیع شده، ایده کلی سیستم عامل‌های توزیع شده نسبتا مشخص تر می شود. سیستم عامل‌های توزیع شده که معمولا براساس Microkernelها طراحی می شوند بر دو یا چند کامپیوتر سوارشده و بااستفاده از همه کامپیوترها به کاربر خود،یک سیستم بزرگ که از درون شبیه به یک سیستم عامل معمول است، ارائه می کنند. در سیستم عامل‌های توزیع شده، هر کامپیوتر به تنهایی معنایی ندارد و قسمتی از بار بزرگ یک سیستم عامل را به دوش می کشد.

محورهای بسیار مهم دراین سیستم عامل‌ها، مخفی بودن ارتباط این کامپیوترها و نحوه توزیع بار سیستم در آنها، استقامت و قابلیت اطمینان این سیستم‌ها،امکانات ساده برای افزایش و کاهش تعداد کامپیوترها ودر نهایت سهولت کاربرد و نگهداری این سیستم‌ها در دراز مدت است.

در این سیستم‌ها پروسه‌ها، فایل‌ها، منابع حافظه مانند دیسک‌های سخت وغیره در سیستم توزیع می شود و سیستم به وجود آمده برآیند تمام امکانات سخت‌افزاری کامپیوترهای عضو مجموعه است. یکی از نمونه‌های این گونه سیستم عامل‌ها، سیستم عامل دانشگاهVrije[1] در آمستردام Amoeba[2] است که به مدت ۵ سال در دانشگاه، صنعت و ارگان‌های دولتی استفاده می‌شده است.

---

1 Amsterdam Vrije university
2 TheAmoebaDistributedOperatingSystem, http://www.cs.vu.nl/pub/amoeba/

۱–۸– مؤلفه های سیستم عامل

سیستم عامل‌ها را می توان به اجزای کوچکتری تقسیم کرد و هر بخش را بطور مستقل بررسی کرد. از آنجا که بررسی تفصیلی این بخش‌ها و الگوریتم‌های پیاده سازی آنها از حیطه این بخش خارج است فقط به آنها اشاره می کنیم.

۱–۸–۱– مدیریت پردازش

هر برنامه از یک یا چند پردازش تشکیل شده است. پردازش‌ها نیاز به منابع سیستمی مانند پردازنده، حافظه و ... دارند و با یکدیگر ممکن است ارتباط داشته باشند. مدیریت این پردازش‌ها و تخصیص منابع به آنها توسط سیستم عامل صورت می گیرد.

۱–۸–۲– مدیریت حافظه اصلی

حافظه اصلی[1] یکی از منابع بسیار پر اهمیت در کامپیوتر است که توسط سیستم عامل مدیریت می شود در حالیکه امروزه یک کامپیوتر معمولی خانگی دو هزار با بیشتر از یک کامپیوتر ۷۰۹۴ قدیمی حافظه اصلی دارد، همچنان برنامه‌ها بزرگتر می شوند و نیاز به حافظه افزایش می یابد. قسمتی که در هر سیستم عامل وظیفه مدیریت حافظه را دارد به مدیر حافظه[2] معروف است و نحوه عملکرد این قسمت تاثیر فراوانی بر عملکرد کلی سیستم عامل دارد.

۱–۸–۳–مدیریت فایل

مدیریت فایل یکی از مشهودترین مؤلفه‌های سیستم عامل می باشد. کامپیوتر می تواند اطلاعات را در انواع مختلف رسانه‌های فیزیکی ذخیره کند. از انجا که هر رسانه ویژگی‌های خاص خود را دارا ست؛ سیستم عامل می باید دید منطقی یکنواختی را از فایل‌ها به کاربر و برنامه‌ها ارائه دهد.

---

1 Main Memory (RAM)
2 Memory Manager

۱-۸-۴- مدیریت سیستم ورودی / خروجی

سیستم عامل‌ها معمولا ورودی‌ها و خروجی های سیستم کامپیوتر را به وسیله مدل‌های نرم‌افزاری بسیار ساده‌تری به کاربر و برنامه‌ها ارائه می کند و اصولا به لحاظ امنیتی به کاربران و برنامه‌ها اجازه نمی دهد تا خود به طور مستقیم با این ورودی‌ها وخروجی‌ها ارتباط برقرار کنند.

۱-۸-۵- مدیریت حافظه ثانوی

کامپیوتر تنها می تواند برنامه هایی که در حافظه اصلی وجود دارند اجرا کند و برنامه‌ها می توانند از اطلاعات خود که در حافظه اصلی هستند استفاده کنند. از آنجایی که حافظه اصلی توانایی ذخیره تمام برنامه‌ها و اطلاعات را ندارد و به لحاظ ساختار با قطع برق اطلاعات آن از بین خواهند رفت، حافظه‌های مجازی به وجود می آیند. سیستم عامل‌ها در کنار سایر وظایف خود مدیریت حافظه ثانوی که معمولا دیسک سخت است را نیز بر عهده دارد.

۱-۸-۶- شبکه

یکی از اجزای سیستم عامل که بخصوص در سیستم عامل‌های توزیع شده بسیار مهم می‌باشد مساله شبکه و مدیریت آن است. در سیستم‌های توزیع شده که بار یک سیستم عامل در چندین کامپیوتر توزیع شده است، در اصل شبکه بستر ارتباط اجزای یک سیستم عامل با یکدیگر است. سیستم عامل‌ها معمولا دسترسی شبکه را به صورت دسترسی یک فایل تعمیم می دهند.

۱-۸-۷-سیستم حفاظتی

وجود چندین کاربر و یا اجرا چندین برنامه در کنار یکدیگر منجر به مسئله‌ی حفاظت می‌شود. سیستم عامل موظف است اطلاعات کاربران را از یکدیگر حفظ کند و حین اجرای برنامه‌ها، برنامه‌ها و اطلاعات آنها را از یکدیگر محفوظ نگه دارد. برای این امر الگوریتم‌های نرم‌افزاری و مکانیزم‌های سخت‌افزاری مختلف تعبیه شده است که در ادامه مفصل‌تر به بررسی آنها خواهیم پرداخت.

۱–۸–۸–مفسر فرمان یا پوسته

ارتباط سیستم عامل با کاربران توسط مفسر فرمان، پوسته یا متشابهات آن به وجود می‌آید. در بعضی از سیستم عامل‌ها مانندMS-DOS و UNIX مفسر فرمان جدا از سیستم عامل است. بعضی از سیستم عامل‌ها،مانند ویندوز و مکینتاش، پوسته‌های مبتنی بر پنجره‌ها و ماوس به کاربر ارائه می کنند که این پوسته‌ها به واسطهای دوستانه با کاربر[1] معروف شده اند.

۱–۹– امنیت سیستم عامل

امنیت یکی از پارامترهای بسیار مهم در هر مجموعه و سیستم است و در کامپیوترها،عمدتا سیستم عامل است که وظیفه برقراری امنیت را در کامپیوتر برعهده دارد. امنیت در دنیای کامپیوتر غالبا در دو محور امنیت در مقابل تهدیدهای خارجی و امنیت داخلی کامپیوتر،بررسی می شود.

در گذشته، سیستم عامل‌ها یا چند کاربره بودند مانند UNIX و یا مانند MS-DOS تک کاربره بودند. در سیستم عامل‌های چند کاربره، امنیت هم در لایه خارجی کامپیوتر اهمیت داشت و هم در داخل خود سیستم عامل و در مقابل کاربران مختلف. در سیستم‌های تک کاربره مانند MS-DOS و Windows 95 غالبا خود کامپیوتر محفوظ تلقی می شد و فقط امنیت درلایه خارجی اهمیت داشت.

تجربه به وجود آمدن ویروس‌های مختلف، کرم‌های اینترنتی و نفوذگرانی که به کامپیوترهای تک کاربره وارد می شدند و با نبودن لایه‌های امنیتی داخلی، به یک باره کل اطلاعات سیستم را به غارت می بردندو صاحب آن سیستم می شدند، نشان داد که اهمیت امنیت داخلی حتی برای سیستم های تک کاربره مانند کامپیوترهای شخصی،از اهمیت امنیت داخلی سیستم‌های چند کاربره کمتر نیست و شرکتهایی که عمدتا سیستم‌های تک کاربره را تولید می کردند (مانند مایکروسافت) مجبور شدند امنیت داخلی سیستم‌ها را نیز مورد توجه قرار دهند.

---

1 User Friendly

۱-۹-۱- خطرات و نقاط ضعف

خطرات زیادی یک کامپیوتر را تهدید می کنند که همگی مبتنی بر نقاط ضعف برنامه‌ها و کامپیوتر هستند. نمونه هایی از نقاط ضعف و خطرات احتمالی در زیر ذکر شده است.

۱-۹-۱-۱- نقاط ضعف برنامه ها[1]

بسیاری از برنامه‌ها بخصوص برنامه‌هایی که برای سرویس دهی در شبکه نگارش می‌شوند، نقطه ضعف‌هایی در نگارش و پیاده سازی دارند، یکی از معروف ترین این نقطه ضعف‌ها پرشدن بافر برنامه یا Stack آنها است.

این برنامه‌ها رفتارهای نابهنجار وغیر قابل پیش‌بینی در این مواقع از خود نشان می دهند که مورد توجه نفوذگران هستند و توسط این نقاط ضعف، نفوذگر قادر به از کار انداختن برنامه یا نفوذ به داخل سیستم، ربودن اطلاعات یا از کار انداختن سیستم می شود.

۱-۹-۱-۲- استراق سمع [2]

تمام اطلاعات ارسالی از طریق شبکه‌ها مانند اینترنت قابل استراق سمع و شنود هستند. این اطلاعات ممکن است اطلاعات آماری بسیار مهم، نامه‌های الکترونیکی، رکوردهای بانک‌های اطلاعاتی و یا نام و رمز عبور کاربران باشد.

یک نفوذگر می تواند با استراق سمع در یک شبکه، به اطلاعات بسیار مهمی دست یابی کند.

۱-۹-۱-۳- خطاها و اشتباهات انسانی

حتی یک سیستم کاملا امن را یک مدیر ارشد یا پرسنل امین آن سیستم با یک خطای انسانی می تواند ناامن سازد.

ارسال رمز عبور سیستم در شبکه توسط پست الکترونیکی و برنامه‌های ارسال پیام یا ذخیره رمز عبور در یک فایل معمولی از جمله خطاهای انسانی هستند.

---

1 Exploits
2 Eavesdropping

۱-۹-۱-۴- از بین بردن سرویس دهی[1]

یک نفوذگر توسط حمله سنگین به کامپیوتر شما و سرویس‌هایی که ارائه می شود باعث بالارفتن بار کلی سیستم و پر شدن خطوط ارتباطی می شود. این حمله که اصطلاحا به DOS معروف است سبب می شود کامپیوتر نتواند سرویس مورد نظر را ارائه کند و نفوذگر به مقصود خود می رسد.

۱-۹-۱-۵- حملات با واسطه (غیرمستقیم)[2]

یک روش بسیار معمول برای نفوذگران استفاده از یک یا چند کامپیوتر واسطه و معمولا بی‌اهمیت که قبلا به آنها نفوذ کرده، برای حمله به یک کامپیوتر مهم است. به این ترتیب نفوذگر معمولا رد حمله خود را مخفی می سازد و از امکانات موجود در آن کامپیوترها مانند عرض باند اینترنت نهایت استفاده را می برد. برای مثال نفوذگر از طریق دستگاه‌های بی اهمیت و کنترل نشده به شبکه شما یا شبکه‌های دیگر نفوذ کرده و از طرق آنها حمله خود را آغاز می‌کند.

۱-۹-۱-۶- در های مخفی[3]

روش‌هایی برای دور زدن از الگوریتم های شناسایی و تطبیق هویت معمولا به درهای مخفی معروف هستند. برای نمونه در بعضی سیستم‌ها خود برنامه سازان برای کنترل سیستم درهای مخفی برای کنترل تولید می کند که گاهی با آشکار شدن توسط هکرها منجر به نفوذهای غیر قابل پیش بینی می شوند. بعضی برنامه‌ها هم برای همین موضوع تولید می شوند و پس از نصب شدن برروی یک کامپیوتر، درهای مخفی برای نفوذگر باز می کنند.

۱-۹-۱-۷- حمله توسط دسترسی مستقیم[4]

تمام کسانی که به یک کامپیوتر دسترسی مستقیم فیزیکی یا حتی دسترسی مناسب تحت شبکه دارند می توانند با حمله به آن کامپیوتر از اطلاعات آن استفاده کنند. نمونه هایی از این

---

1 Denial of Service
2 Indirect Attacks
3 Back Doors
4 Direct Access Attacks

حمله‌ها نصب نسخه‌های تغییر یافته سیستم عامل، نصب برنامه‌ها یا دستگاه‌های ثبت کننده کلیدهای فشرده شده صفحه کلید[1]و یا باز کردن دیسک سخت و کپی اطلاعات موجود درآن است.

۱-۹-۲- راه کارهای نرم‌افزاری مقابله با حملات در سیستم عامل

در مقابل خطرات نفوذگران، راهکارهایی برای افزایش ضریب امنیت وجود دارد که شرح مختصری از آنها در زیر آورده شده است.

۱-۹-۲-۱- ابزارهای کنترل الگوریتم[2]

برای کنترل الگوریتم‌های که برای سیستم‌های امن نگارش می شوند، الگوریتم‌های کنترل اتوماتیک وجود دارد که قادر به آشکارکردن بعضی از نقاط ضعف در الگوریتم‌ها و سیستم‌ها هستند.

۱-۹-۲-۲- استفاده از سیستم عامل‌های با ساختار Micro Kernel

سیستم عامل‌هایی که با ساختار سیستم Micro Kernel نگارش می شوند کد بسیار کمتری در فضای کاربر ارشد یا مد سیستم اجرا می‌کنند و بسیاری از عملیات مربوط به سیستم عامل در فضای کاربر عمومی سیستم نگارش می شود.

دراین سیستم‌ها نفوذگر حتی اگر بتواند به قسمتی از فضای خود سیستم عامل نفوذ کند نمی تواند کل سیستم را تحت تاثیر قرار دهد.

۱-۹-۲-۳- رمزنگاری و پنهان کردن اطلاعات[3]

استفاده از الگوریتم‌های رمزنگاری در قسمت‌های مختلف می تواند از بسیاری حملات جلوگیری کند.

---

1 Key Logger
2 Automated Theorem Proving
3 Cryptgraphic

برای مثال ارسال اطلاعات رمز شده در شبکه به جای اطلاعات معمول، می‌تواند جلوی حملاتی مانند استراق سمع را بگیرد و شنود نفوذگر را پوشش دهد. رمز کردن نام و رمز عبور در سیستم‌ها نیز منجر می شود تا نفوذگر نتواند رمز کاربر ارشد را استخراج کند.

۱-۹-۲-۴- تکنیک های شناسایی قوی [1]

برای شناسایی کاربران، بخصوص کاربران راه دور که از دور نام و رمز عبور خود را به سیستم ارائه می‌کنند، الگوریتم‌ها و تکنیک‌های معتبر لازم است. وجود این تکنیک‌ها از لورفتن اطلاعات در طول راه جلوگیری می کند.

۱-۹-۲-۵- تکنیک های برنامه های معتمد[2]

در این روش فقط از برنامه‌هایی که نگارنده آنها، امنیت لازم را در برنامه خود تضمین می‌کنند واین برنامه‌ها بطور مخصوص علامت خورده اند،استفاده می‌شود. لازم به ذکراست که در دنیای نرم‌افزاری بعضی برنامه‌ها و بخصوص برنامه های حیاتی که ممکن هستند سلامتی انسان‌ها را به خطر بیندازند (مانند برنامه‌های دستگاه‌های پزشکی)،دارای تضمین‌های مختلف از شرکت نگارنده هستند و یک سیستم عامل می تواند به نحوی تنظیم شود که فقط این برنامه‌ها را اجرا کند.

۱-۹-۲-۶- کنترل دسترسی اجباری۳

در این تکنیک حیطه اختیار هر کاربر یا برنامه بطور خاص تعریف می شود و سیستم عامل اجازه نمی دهد تا یک کاربر یا برنامه از حیطه خود خارج شود. این تکنیک قبلا در سیستم عامل‌های دولتی و نظامی وجود داشت و به تازگی در سیستم عامل‌های متداول و پر مصرف مانند لینوکس نیز پیاده سازی شده است.

---

1 Strong Authentication Techniques
2 Chain of Trust Techniques
3 Mandatory Access Control

۱-۹-۲-۷- تکنیک توانایی و تکنیک لیست دستیابی[1]

این تکنیک‌ها برای دادن اختیار و یا محدود کردن آن برای کاربران و برنامه‌ها از طریق سیستم عامل است. دراین تکنیک‌ها سعی بر محدود کردن اختیار کاربر یا برنامه در مناطق مختلف کامپیوتر و در نهایت ایجاد امنیت در سیستم‌های چند کاربره است.

۱-۹-۳- روش های مقابله و پیش گیری متفرقه

روش‌های زیر گرچه در مقوله این مبحث نمی گنجد، ولی به علت اهمیت آن در اینجا ذکر شده است و جنبه عمومی دارند

۱-۹-۳-۱- تحقیق قبل از استفاده از برنامه

بعضی از سایت‌های اینترنتی بطور رایگان لیستی از اشکالات و منافذ امنیتی را در برنامه‌های مختلف ارائه می‌کنند. در صورتیکه از برنامه خاصی لازم است استفاده شود حتی المقدور باید قبل از استفاده تحقیق لازم درباره آن برنامه صورت گیرد. یکی از نمونه‌های این سایت‌ها، www.secunia.com است.

۱-۹-۳-۲- پشتیبان گیری

گرفتن پشتیبان بطور متوالی از اطلاعات حیاتی یکی از کارهای بسیار مهم و در واقع چاره پس از وقوع حادثه‌های مختلف مانند خرابکاری اطلاعاتی و خرابی‌های متفرقه سخت‌افزار است. روش‌های متداول استفاده از لوح‌های فشرده و نوارهای مغناطیسی است که طول عمر بالا دارند و مقرون به صرفه هستند.

۱-۹-۳-۳- استفاده از نرم‌افزارهای ویروس گیری

نرم‌افزارهای بسیاری در بازار وجود دارند که همگی نقش شناسایی و حذف ویروس را دارند. استفاده از این نرم‌افزارها در کنار سایر راهکارهای امنیتی به شدت توصیه می شود تا در صورت ورود ویروس،این برنامه‌ها شناسایی شوند.

---

1 Capability and Access Control List (ACL)

۱-۹-۳-۴- دیوارهای آتش[1]

برنامه‌های دیوار آتش و سخت‌افزارهای موجود در این زمینه از ورود نفوذگران از طریق کانال‌ها و پورت‌های باز جلوگیری می کند.

۱-۹-۳-۵- سیستم شناسایی ورود غیر مجاز[2]

این سیستم با کنترل اطلاعات موجود در شبکه، از تلاش برای نفوذهای افراد غیر مجاز آگاه می شود. برای مثال در صورتیکه یک فرد یا برنامه بخواهد با سیستم سعی و خطا و با ارسال بی شمار نام کاربری و رمز عبور به یک سیستم نفوذ کند، این سیستم آگاه شده و می‌تواند به مدیر شبکه یا برنامه دیوار آتش اطلاع دهد تا از نفوذ این فرد جلوگیری به عمل آید.

۱-۹-۳-۶- هوشیاری انسانی

در صورتیکه مدیران و کاربران سیستم از نام‌ها و رمزهای عبور به خوبی حفاظت کنند و بطور مثال آنها را به صورت الکترونیکی در شبکه ناامن منتشر نکنند یا آنها را درمحل ناامن ذخیره نکنند و در انتخاب رمز عبور خود از توصیه های لازم استفاده کنند، تعداد بی شماری از نفوذها خود به خود خنثی خواهد شد.

۱-۱۰-نتایج

در این قسمت مبانی سیستم و نکات کلیدی آن بررسی شده است. باتوجه به مطالب مذکور ویژگی‌های مطلوب برای یک سیستم عامل متن باز Enterprise مشهوداست که در ادامه نکات کلیدی این ویژگی‌ها مجددا دسته بندی شده است:

۱- دارا بودن تمام ویژگی‌های سیستم عامل‌های کلاسیک (اشتراک زمانی، مدیریت پردازش‌ها، مدیریت حافظه، مدیریت فایل و...)

۲- امکان انتقال سیستم عامل به سخت‌افزارهای مختلف مانند x86 ، RISC ، PowerPC و غیره.

---

1 Firewalls
2 Intrusion-Detection Systems (IDS)

۳- انطباق ساختار داخلی سیستم عامل با استانداردهایی نظیر POSIX به جهت انتقال برنامه‌ها در لایه متن اولیه

۴- امکان توزیع سیستم عامل و برنامه‌ها در چند کامپیوتر

۵- پشتیبانی از چند پردازندگی و سیستم‌های موازی

۶- امکانات قوی برای امنیت و وجود لایه های امنیت داخلی و خارجی

بعضی ویژگی‌های که در محدوده این بحث نبوده؛ در این قسمت به آنها بطور مختصر اشاره شده است:

۱- پایداری و خوشنامی سیستم عامل

۲- دارا بودن مجوزهای متن باز لازم برای استفاده درایران

۳- پشتیبانی گسترده و بروز توسط تیم‌ها یا شرکت های معتبر

۴- پشتیبانی چند زبانی

۵- واسط کاربری قوی و حتی المقدور سازگار یا قابل انطباق جهت پشتیبانی زبان فارسی

فصل دوم – بررسی و مقایسه سیستم عامل‌های متن باز موجود

امروزه تعداد کثیری از نرم‌افزارهای سیستم‌عامل متن باز وجود دارند که در این قسمت سعی شده تا با بررسی مهم‌ترین عناوین این نرم‌افزارها به قابلیت‌های آنها اشاره شود و با مقایسه کلی، بهترین گزینه‌ها انتخاب شوند.

در این قسمت ابتدا به تاریخچه، معرفی و بررسی سیستم عامل‌های متن باز پرداخته شده و سپس امکانات فنی و خصوصیات آنها با یکدیگر در جداول مختلف مقایسه و بررسی شده‌اند.

دقت شود که بهترین سیستم‌عامل بطور کلی وجود ندارد و هر سیستم عاملی می‌تواند در یک محیط خاص مانند یک پروژه به دلایل خود، بهترین انتخاب باشد.

۲–۱– دسته بندی سیستم عامل‌های متن باز

سیستم‌عامل‌های متن باز را می‌توان در یک تقسیم‌بندی ابتدا به دو دسته سیستم‌عامل‌های تحقیقاتی/ علمی و سیستم‌عامل‌های عملیاتی / کاربردی تقسیم کرد. به علت کثرت سیستم عامل‌های متشابه یونیکس، دسته‌های فوق را می‌توان هریک به دو دسته سیستم‌عامل‌های متشابه یونیکس و سیستم‌عامل‌های غیر متشابه یونیکس تقسیم کرد که کلا در زیر در چهار دسته عنوان شده‌اند.

۲–۱–۱– سیستم عامل‌های آکادمیک و تحقیقاتی متشابه با یونیکس

مهم‌ترین و معروف‌ترین سیستم عامل‌هایی که دراین دسته جا دارند عبارتند از: Unix ، Minix ، Plan 9 ، Inferno ، Plan B ، Xinu و Solaris. این سیستم‌عامل‌ها اکثرا یا جنبه تحقیقاتی و علمی دارند و یا بیشتر جهت آموزش در دانشگاه‌ها استفاده می‌شوند. البته در میان این سیستم عامل‌ها، سیستم عامل Solaris چندان جنبه آکادمیک ندارد ولی به جهت داشتن کد اصلی سیستم عامل SVR4 که مدت‌ها در دانشگاه‌ها تدریس می‌شده است، در این رده قید شده است.

۲–۱–۲– سیستم عامل‌های عملیاتی متشابه با یونیکس

مهم‌ترین و معروف‌ترین سیستم عامل‌هایی که دراین رده جای دارند عبارتنداز: OpenSolaris،خانوادهBSD شاملFreeBSD،OpenBSDوNetBSD، Linux و OpenDarwin.

این سیستم‌عامل‌ها، سیستم‌عامل‌های پرمصرف و کاربردی هستند که کد منبع آنها باز است واکثرا با استاندارد POSIX منطبق هستند.

۲-۱-۳- سیستم عامل‌های آکادمیک و تحقیقاتی غیر متشابه با یونیکس

مهم‌ترین و معروف‌ترین سیستم‌عامل‌هایی که در این رده جا دارند عبارتنداز :Mach، Nemesis، V، L4، ILIOS، House، Coyotos و Amoeba. این سیستم‌عامل‌ها عمدتا یا دارای طرح‌ها علمی جدید هستند و یا جنبه آکادمیک دارند ودر محیط‌های عملیاتی آنچنان استفاده نمی‌شوند.

۲-۱-۴- سیستم عامل‌های عملیاتی غیرمتشابه با یونیکس

مهم‌ترین و معروف‌ترین سیستم‌عامل‌هایی که در این رده جادارند عبارتند از :ReactOS، FreeDOS و Haiku. این سیستم‌عامل‌ها همگی به ترتیب الگو برداری از سیستم‌عامل‌های معروفی چون Windows NT ، MS-DOS و BeOS هستند که در ادامه بیشتر به آنها خواهیم پرداخت.

به دلایل واضح، سیستم‌عامل Enterprise انتخابی ما باید از سیستم‌عامل‌های عملیاتی باشد و به دلایل مختلف از جمله کثرت و خوشنامی سیستم‌عامل‌های متشابه یونیکس، در این رده قرار گیرد. متشابه بودن سیستم‌عامل انتخابی ما با یونیکس و منطبق بودن آن با استانداردهایی چون POSIX به ما این امکان را می‌دهد که از بسیاری نرم‌افزارهای متن باز دیگر که از این استاندارد حمایت می‌کنند استفاده کنیم و در لایه کد اولیه، آنها را به سیستم‌عامل مورد نظر خود منتقل کنیم.

۲-۲- بررسی سیستم عامل‌های معروف

مهم‌ترین سیستم‌عامل‌های معروف و پرمصرف متن‌باز و عملیاتی امروزی به طور مختصر در زیر بررسی شده‌اند. در این میان تاریخچه سیستم‌عامل UNIX به‌عنوان یکی از مهم‌ترین سیستم‌عامل‌های اولیه و در کنار آن، سیستم‌عامل BSD که هر دو نقش پدران بسیاری از سیستم‌عامل‌های امروزی را دارند، نیز آورده شده است.

۲-۲-۱- تاریخچه سیستم عامل یونیکس و BSD

یونیکس در دهه‌های ۱۹۶۰ و۱۹۷۰ توسط AT&T Bell Labs نگارش شده است. اهداف اولیه یونیکس، قابلیت حمل آن به سخت‌افزارهای مختلف[1]، چند تکلیفی[2]، چند کاربری[3] واشتراک زمانی[4] بود.

شکل ۲-۱: آرم سیستم عامل Unix

از خصوصیات سیستم‌عامل یونیکس، فایل‌های متنی ساده، مفسر فرمان خطی، ساختار فایل سلسله مراتبی[5] و پیاده‌سازی اجزاء سخت‌افزار کامپیوتر به صورت فایل‌ها می‌باشد. در مهندسی نرم‌افزار، سیستم‌عامل یونیکس بخاطر کاربرد گسترده زبان برنامه‌سازی C و تکنیک‌های آن، معروف شده است. زبان C و یونیکس هر دو توسط AT&T خلق و توسعه پیدا کرده‌اند و در دولت و مراکز پژوهشی و دانشگاه‌ها توزیع شدند.

در حال حاضر مالکیت عنوان تجاری یونیکس متعلق به Open Group است و کد منبع اصلی این سیستم‌عامل متعلق به SCO Group و Novel است. سیستم‌عامل‌هایی که با استاندارد Single Unix Specification (تنها توصیف یونیکس) منطبق هستند، صلاحیت عنوان یونیکس را دارند و می‌توانند از این نام استفاده کنند. سیستم‌عامل‌های نزدیک به این استاندارد، به سیستم‌عامل‌های متشابه با یونیکس معروفند.

سیستم‌عامل‌های یونیکس امروزی هم در کاربرد رومیزی[6] و هم در کاربرد سرور بسیار استفاده می‌شود. ساختار این سیستم‌عامل و مدل Client/Server این سیستم‌عامل، در گذشته نقش بسزایی در رشد و توسعه اینترنت داشته است.

---

1 Portable
2 Multi task
3 Multi User
4 Time Sharing
5 Hierarchical File System
6 Client

تصویر آورده شده برای این بخش، از سایت unix.org انتخاب شده و شعار معروف این وب سایت است. متن داخل تصویر اشاره به زندگی آزاد (قدرت انتخاب و عدم انحصار طلبی) می‌کند.

در کنار سیستم‌عامل یونیکس اولیه، سیستم‌عاملی به نام BSD وجود دارد که مبتنی‌بر نگارش‌های اولیه یونیکس بوده است. سیستم‌عامل‌های اولیه یونیکس به دانشگاه‌ها اجازه می‌دادند تا کد منبع این سیستم‌عامل را تغییر دهند و آن را کامل کنند. تدوین و نگارش سیستم‌عامل BSD بر همین اساس در دانشگاه برکلی در دهه ۱۹۷۰ شروع شد و این سیستم‌عامل در ابتدا به صورت توسعه‌ها و نرم‌افزارهایی برای نسخه ششم سیستم‌عامل AT&T UNIX بود. بسیاری از تکنولوژی‌های امروزی شبکه مانند IP[1] به تدریج در نسخه‌های این سیستم‌عامل به وجود آمدند.

با گذشت زمان و کامل شدن نسخه‌هایBSD، این سیستم عامل به یک سیستم‌عامل مطرح روز تبدیل گشت. در همین دوران به علت نوع لیسانس UNIX و در نهایت شکایات قانونی AT&T نسبت به وجود فایل‌های کد منبع UNIX در کد منبع سیستم‌عامل BSD، به تدریج در طول چندین سال سیستم‌عامل BSD در دانشگاه برکلی و توابع آن از نو نگارش شد و در کد منبع جدید، از کد منبع موجود در سیستم‌عامل AT&T استفاده‌ای نشد. به این ترتیب مشکلات قانونی کد منبع این سیستم‌عامل به تدریج حل شده و این سیستم‌عامل به اوج قدرت خود رسید.

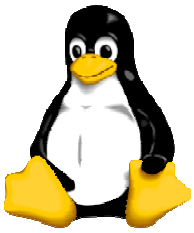
کد منبع جدید تولید شده با لیسانس BSD ارائه می‌شد ومبنای رشد بسیاری از سیستم‌عامل‌های متن باز و خصوصی قرار گرفت. این سیستم‌عامل به تدریج در دهه ۱۹۹۰ عمدتا توسط سیستم‌عامل‌های متن بسته SVR4 و OSF/1 که از کد منبع همین سیستم‌عامل نیز استفاده می‌کردند، جایگزین شد. امروزه به وفور از نسخه‌های متن باز و مدرن‌تری که از این سیستم‌عامل منشق شده‌اند، استفاده می‌شود و نام BSD در حال حاضر به سیستم عامل‌های مدرن امروزی که از همین سیستم عامل BSD شروع شدند (مانند FreeBSD و NetBSD) نیز اطلاق می‌شود.

---

[1] Internet Protocol

۲-۲-۲- سیستم عامل لینوکس

در سال ۱۹۸۳ ریچارد استالمن، پروژه GNU را تاسیس کرد. هدف استالمن تولید و توسعه یک سیستم‌عامل کامل متشابه با یونیکس بر اساس مدل نرم‌افزارهای متن باز بود. در اوایل دهه ۱۹۹۰ بیشتر اجزای یک سیستم‌عامل توسط این پروژه جمع‌آوری یا تولید شده بود و مساله اصلی هسته سیستم عامل (Kernel) بود. پروژه GNU در سال ۱۹۹۰ شروع به تولید هسته خود به نام Hurd بر اساس Mach Microkernel کرد، اما تولید این هسته بر اساس Mach روالی بسیار کند داشت و با مشکلات متعدد همراه بود.



شکل ۲-۲: آرم سیستم عامل لینوکس

در سال ۱۹۹۱ یک هسته جدید که در اصل سرگرمی یک دانشجوی فنلاندی در دانشگاه Helsinki به نام لینوس توروالدس بود،به وجود آمد. این دانشجو در کامپیوتر شخصی خود از سیستم‌عامل Minix نگارش پروفسور آندرو تننبام استفاده می‌کرد. سیستم‌عامل Minix اصولا برای تدریس تولید شده بود و آقای آندرو تننبام سعی می‌کرد تا آنجا که امکان داشت،سیستم‌عامل Minix را کوچک نگه دارد تا برای ارائه در کلاس درس ممکن باشد و از این رو اجازه نمی‌داد تا کسی سیستم‌عامل Minix را توسعه بدهد. این امر سبب شد تا توروالدس سیستم‌عامل جدید خود را به عنوان یک سرگرمی آغاز کند. آقای توروالدس دراصل نام Freax را از ترکیب واژه‌های free و freak و ترکیب حرف X از Unix برای اولین نگارش سیستم‌عامل خود انتخاب کرده بود اما آری لماک که مدیریت یکی از ftp serverهای دانشگاه را برعهده داشت، نام دایرکتوری Linux را برای سیستم‌عامل و فایل‌های سیستم‌عامل آقای توروالدس انتخاب کرد.

در نسخه‌های اولیه لینوکس، از سیستم‌عامل Minix به عنوان یک بستر برای کامپایل فایل‌ها و هسته لینوکس استفاده می‌شد و لینوکس حتی دارای یک روال Boot مناسب و

منسجم نبود. برای کار با لینوکس می‌باید سیستم‌عامل دیگری مانند Minix وجود داشت تا روال Boot را انجام داده و سپس لینوکس را مانند یک برنامه آغاز کند. با گذشت زمان نواقص لینوکس به سرعت تکمیل شدند و برای مثال برنامه‌هایی مانند LILO به سرعت به‌عنوان برنامه Boot کننده لینوکس به وجود آمدند.

طولی نکشید که لینوکس از Minix در عملکرد پیش افتاد و توروالدس و برنامه‌سازان هسته اولیه این سیستم‌عامل، کار خود را با پروژه GNU تطبیق دادند تا بتوانند از ابزارها و برنامه‌های تولید شده در این پروژه نهایت استفاده را بکنند و یک سیستم‌عامل کامل را به وجود آورند.

حرکت‌ها و رشدهای سریع سیستم‌عامل لینوکس طی سال‌های گذشته جزو داغ‌ترین اخبار در دنیای فن‌آوری اطلاعات و سیستم‌عامل‌ها بوده و امروزه این سیستم‌عامل به یکی از کاملترین و محبوب‌ترین سیستم‌عامل‌ها تبدیل شده است و همچنان رشد این سیستم‌عامل ادامه دارد.

اگرچه سیستم‌عامل لینوکس در کل به یک سیستم‌عامل با امکانات امنیتی مناسب معروف است ولی پروژه‌ای به نامSELINUX در سال۲۰۰۰ توسط سازمان آمریکایی National Security Agency به جامعه متن باز ارائه شد که امنیت را در این سیستم‌عامل بسیار افزایش داده است. این پروژه یک پیاده‌سازی سیستم امنیتی به نام سیستم‌کنترل دستیابی اجباری (MAC)[1] برای هسته لینوکس است و به‌عنوان یک ارزش افزوده برای این سیستم عامل در بین سیستم عامل‌های دیگر مطرح است. امروزه این پروژه به دلایل اهمیت فراوان، حتی به سیستم‌عامل‌های دیگر مانند FreeBSD هم انتقال یافته است. در حال حاضر متداول‌ترین روال‌های امنیتی لینوکس عبارتند از:۱- اجازه دسترسی به فایل‌ها[2] ۲- لیست‌های کنترل دستیابی (ACL) و ۳- سیستم SELinux که کلا لینوکس را به یک سیستم‌عامل مناسب و امن تبدیل می‌کنند.

سیستم‌عامل لینوکس به علت ماهیت متن باز خود، تعداد نسبتا فراوانی از فایل سیستم‌ها را نیز پشتیبانی می‌کند. مهم‌ترین این فایل سیستم‌ها، Ext3، JFS، Reiser و XFS است. اگرچه این فایل سیستم‌ها در پیاده‌سازی با هم اختلافاتی دارند ولی برای کاربردهای روزمره، نسخه-

1 Mandatory Access Control
2 File system permissions

های به روز این فایل سیستم‌ها آنچنان اختلافی با یکدیگر ندارند و بسیاری از مدیران ارشد سیستم‌ها، به فایل سیستم پیش فرض توزیع کننده لینوکس خود که معمولا یا Ext3 و یا Reiser است، اعتماد می‌کنند. به علت تغییر و رشد سریع امکانات این فایل سیستم‌ها، در کاربردهای خاص و پراهمیت، قبل از انتخاب یکی از این فایل سیستم‌ها، بهتر است مدیران ارشد و طراحان پروژه، آخرین وضعیت این فایل سیستم‌ها را با نیازهای پایه‌ای پروژه خود کنترل و مقایسه کنند.

۲-۲-۳- سیستم عامل FreeBSD

سیستم‌عامل FreeBSD به ترتیب از سیستم‌عامل‌های AT&T Unix و سپس BSD و در نهایت از کد 386BSD و 4.4BSD منشق شده است. این پروژه در سال ۱۹۹۳ کد اولیه خود را از 386BSD برداشت و بخاطر دعوای قانونی Novell و Berkeley سر مالکیت یونیکس بیشتر کد خود را مجددا به کمک 4.4BSD Lite نگارش کرد و در سال ۱۹۹۵ نسخه FreeBSD 2.0 خود را ارائه داد.

شکل ۲-۳: آرم سیستم عامل freeBSD

از مزیت‌های نسخه FreeBSD 2.0 سیستم مدیریت حافظه آن بود که از پروژه هسته Mach اقتباس شده بود. این مدل مدیریت حافظه به شدت برای سیستم‌عامل‌هایی که زیر بار سنگین هستند، بهینه شده بود. مزیت دیگر موجود در سیستم‌عامل FreeBSD 2.0 به وجود آمدن شاخه ports بود که امکان دریافت، کامپایل و نصب برنامه‌های ثالث را از اینترنت به سهولت قرار می‌داد. سیستم‌عامل FreeBSD در سایت‌های Hotmail.com،Cdrom.com و Yahoo.com که کاربران زیادی دارند و بار شبکه سنگین دارند، استفاده می‌شد و به آنها قدرت می‌داد.

نسخه FreeBSD 3.0 پشتیبانی از فایل‌های دودویی و اجرایی ELF را نیز در بر داشت و به صورت اولیه از SMP و سیستم‌های ۶۴ بیتی Alpha پشتیبانی می‌کرد. شاخه FreeBSD3.X و FreeBSD4.X بسیاری از مسائل سیستم‌عامل و برنامه‌های جانبی را بهینه کردند.

شاخه FreeBSD5.X پشتیبانی بهتری از سیستم‌های SMP، چند ریسمانی و لایه دسترسی ورودی/خروجی سیستم‌عامل داشت و سیستم‌عاملی بسیار پایدار بود. سیستم‌عامل‌های FreeBSD 6 و FreeBSD 7 که هم اکنون در حال توسعه و نگارش هستند، در حال افزودن امکانات بیشتر برای SMP و استاندارد 802.11 هستند.

در کنار تیم اصلی FreeBSD پروژه‌ای به نام TrustedBSD توسط روبرت واتسون مدیریت می‌شود که هدف آن افزودن تکنیک‌های امنیتی به سیستم‌عامل FreeBSD است. بسیاری از دستاوردهای این پروژه به تدریج به هسته سیستم‌عامل FreeBSD اضافه شده است. از اهداف این پروژه اضافه کردن امکانات ACL یا لیست‌های کنترل دستیابی[1]، ویژگی امنیتی، کنترل دسترسی[2] اجباری، انتقال الگوریتم‌های FLASK/TE موجود در SELinux، تکنیک‌های OpenBSM شرکت Sun و غیره است. این پروژه سیستم‌عامل FreeBSD را به یکی از امن‌ترین سیستم‌عامل‌های موجود تبدیل خواهدکرد و چون کد منبع این پروژه BSD است، خروجی‌های این پروژه به تدریج در OpenBSD و OpenDarwin هم ظاهر خواهند شد.

سیستم‌عامل FreeBSD هم‌مانند Linux، از فایل سیستم‌های متعددی پشتیبانی می‌کند ولی تعداد آنها در مجموع کمتر از Linux است. سیستم‌عامل FreeBSD بطور پیش فرض، از فایل سیستم UFS2 استفاده می‌کنند که فایل سیستمی بسیار پایدار و مطمئن است.

۲-۲-۴- سیستم عامل NetBSD

سیستم‌عامل NetBSD مانند FreeBSD از 386BSD و Net/2 BSD شروع شده است. چهار موسس اولیه NetBSD سیستم‌عامل NetBSD خود را براساس دو خاصیت: ۱-امکان حمل و نقل، ۲-کد برنامه تمیز و بدون اشکال، قرار دادند. اولین نسخه عمومی این سیستم‌عامل در سال ۱۹۹۳ ارائه شده است.

---

1 Access Control List
2 Mandatory Access Control

شکل ۲-۴: آرم سیستم عامل Net BSD

سیستم‌عامل NetBSD 1.0 در سال ۱۹۹۴ ارائه شد و چندین نوع سخت‌افزار متفاوت منجمله PC، HP 9000، Amiga، 68K و غیره را پشتیبانی می‌کرد. در همین سال یکی از چهار موسس این سیستم‌عامل به نام تئو د رات پس از مشاجرات مختلف با سه تن دیگر از این پروژه جدا شد و در سال ۱۹۹۵ پروژه‌ای به نام OpenBSD را آغاز کرد.

در سال ۱۹۹۸ نسخه۱.۳ این پروژه عرضه شد و در سال۱۹۹۹ نسخه ۱.۴ این پروژه ۱۶ نوع سخت‌افزار مختلف را پشتیبانی می‌کرد. در سال ۲۰۰۴ نسخه۲.۰ این سیستم‌عامل ارائه شد که تفاوت عمده آن در پشتیبانی از SMP و چند ریسمانی در سخت‌افزارهای گوناگون بود. نسخه ۲.۰ این سیستم‌عامل از ۴۸ نوع سخت‌افزار مختلف بطور کامل پشتیبانی می‌کرد و از ۶ نوع سخت‌افزار دیگر نیز در سطح کد منبع[1] حمایت می‌کند.

در حال حاضر این سیستم‌عامل به طور گسترده‌ای به سخت‌افزارهای مختلف حمل شده‌است و بیش از ۵۴ نوع سخت‌افزار و ۱۷ نوع پروسسور را می‌شناسد. شعار این سیستم‌عامل این است : "قطعا NetBSD در سخت‌افزار شما اجرا می‌شود. "

این سیستم‌عامل به لحاظ پشتیبانی گسترده از انواع سخت‌افزارها و نوع لیسانس کد منبع آن، بخصوص برای کاربرد در کامپیوترهای تعبیه شده بسیار مناسب است. این سیستم‌عامل رشدی بسیار کند و با دقت دارد. نسخه‌های این سیستم‌عامل بسیار مطمئن هستند و با وجود اینکه این سیستم‌عامل با هدف پشتیبانی گسترده از تمام سخت‌افزارها طراحی می‌شوند، نسخه‌های جدید این سیستم‌عامل بازده بسیار مناسبی را در کاربردهای سنگین ارائه می‌دهند.

سیستم‌عامل NetBSD مشابه سیستم‌عامل FreeBSD از فایل سیستم‌های متعددی پشتیبانی می‌کند که مهم‌ترین آن UFS2 است. این سیستم‌عامل در کل تکنیک‌های امنیتی

---

1 Source Code

کمتری را نیز نسبت به FreeBSD و لینوکس دارد و از امکانات:۱- اجازه دسترسی به فایل‌ها و ۲- تکنیک Veriexec پشتیبانی می‌کند.

۲-۲-۵- سیستم عامل OpenBSD

همان‌طور که گفته شد در سال ۱۹۹۴ یکی از چهار موسس سیستم‌عامل NetBSD به‌نام تئو درات پس از مشاجرات مختلف با سه تن دیگر از این پروژه جدا شد و در سال ۱۹۹۵ پروژه‌ای به نام OpenBSD را آغاز کرد. موسس این سیستم‌عامل به اصرار در استفاده از نرم‌افزارهای متن باز، مستندسازی گسترده و عدم تطبیق با نرم‌افزارهای متن بسته و خصوصی معروف است. او همچنین در سطح برنامه‌سازی به افزون‌سازی مسائل امنیتی و تصحیح‌های مختلف کد منبع معروف است. این پروژه از منزل شخصی تئو درات در کانادا هدایت می‌شود. آقای لینوس تروالدس مدیر پروژه OpenBSD را انسانی سخت توصیف کرده است.

شکل ۲-۵: آرم سیستم عامل Open BSD

سیستم‌عامل OpenBSD دارای خصوصیات و لایه‌های امنیتی شدید است که معمولا یا در سیستم‌عامل‌های دیگر حضور ندارند و یا بطور انتخابی هستند و خصوصیات امنیتی این سیستم‌عامل و لیسانس استفاده از این سیستم‌عامل سبب شده تا تعداد قابل توجهی از محصولات تجاری ازجمله سیستم‌های تعبیه شده مورد استفاده در شبکه‌ها مانند دیوارهای آتش، براساس این سیستم‌عامل به‌وجود آیند. این سیستم‌عامل در حال حاضر بر ۱۶ نوع سخت‌افزار مختلف اجرا می‌شود.

در گزارشات مختلف آمده است که سیستم‌عامل OpenBSD در بعضی کاربردها نسبت به سایر اعضای خانواده BSD ،اندکی کندتر پاسخ می‌دهد. متخصصین سیستم عامل OpenBSD، این امر را بخاطر وجود لایه‌های امنیتی این سیستم‌عامل می‌دانند و هدف اولیه این سیستم‌عامل را امنیت عنوان می‌کنند تا بازدهی سریع سیستم‌عامل.

شعار این سیستم‌عامل در وب سایت خود تا مدت‌ها جمله: "در طول ۶ سال با تنظیمات اولیه، هیچ حفره امنیتی قابل نفوذ از خارج سیستم‌عامل کشف نشده‌است"، بود تا اینکه در

سال ۲۰۰۲ مارک دود از Internet Security Systems یک اشکال در کتابخانه OpenSSH پیدا کرد. این کتابخانه تقریبا در تمام سیستم‌عامل‌های متن باز امروزی استفاده می‌شود و با یافت شدن این اشکال بسیاری از سیستم‌عامل‌ها قابل نفوذ از دنیای خارج بودند. پس از پیدا شدن این حفره امنیتی، شعار این سیستم‌عامل در حال حاضر به جمله: "پس از ۸ سال فقط یک حفره امنیتی برای نفوذ خارجی در تنظیمات اولیه کشف شده است" تغییر کرد و این داستان سبب محبوبیت پیش از پیش این سیستم‌عامل شد.

گرچه سیستم‌عامل OpenBSD در محافل مختلف به سیستم‌عاملی بسیار امن توصیف می‌شود، ولی این سیستم‌عامل تنها از تکنیک عمومی اجازه دسترسی به فایل‌ها که تقریبا در تمام نگارش‌های یونیکس وجود دارد برای کنترل منابع خود استفاده می‌کند و در عوض تمرکز بیشتری در بهینه‌سازی هسته و برنامه‌ها دارد.

۲-۲-۶- سیستم عامل OpenSolaris

سیستم‌عامل OpenSolaris نسخه متن باز سیستم‌عامل Sun Solaris است. شرکت Sun در اوایل سال ۲۰۰۴ پروژه متن باز کردن سیستم‌عامل Solaris خود را آغاز کرد و در سال ۲۰۰۵ عمده کد منبع این سیستم‌عامل را توسط وب سایت OpenSolaris.org تحت لیسانس CDDL که مشابه لیسانس MPL است، ارائه کرد. البته قسمت کوچکی از کد منبع این سیستم‌عامل به صورت متن بسته باقی‌مانده است.

opensolaris

شکل ۲-۶: آرم سیستم عامل Open Solaris

سایت سیستم‌عامل OpenSolaris بیش از ۱۱،۰۰۰ عضو فعال خارج از شرکت Sun دارد و تصور می‌شود که این افراد همچنان اضافه شوند. خالق لینوکس، آقای توروالدس چندان نظر مساعدی نسبت به این سیستم‌عامل ندارد و آن را نسخه لنگ سیستم‌عامل Solaris می‌پندارد، ولی در کل اذعان می‌کند که این سیستم‌عامل طرفداران زیادی دارد که او با آنها به سختی رقابت می‌کند و در نتیجه، برای آنها آرزوی مرگ دارد !

سیستم عامل OpenSolaris فقط یک سیستم‌عامل نیست و در بسته نرم‌افزاری این سیستم‌عامل، مجموعه‌های بسیار کاملی از ابزارهای مختلف نرم‌افزاری گنجانده شده‌است که این سیستم‌عامل را به یک محیط بسیار کامل و هماهنگ برای اهداف مختلف تبدیل می‌کند.

برای نمونه در حال حاضر شرکت Sun Microsystems در کنار سیستم‌عامل خود، برنامه‌های Application Server و Development Tools و بسیاری از برنامه‌های دیگر که سابقاً همگی به صورت تجاری در قبال اخذ هزینه عرضه می‌شدند را به رایگان به کاربران خود ارائه می‌کند.

امروزه عرضه رایگان ابزارهای نرم‌افزاری و خود سیستم‌عامل OpenSolaris توسط شرکت Sun در قالب چند CD رایگان و قابل دریافت از اینترنت،(در حالیکه چند ماه قبل همه این برنامه‌ها و ابزارها، بهایی گران‌قیمت داشتند و داشتن آنها حتی در خواب هم برای بعضی شرکت‌ها قابل تصور نبود.) از عوامل محبوبیت و استقبال این سیستم‌عامل به شمار می‌روند.

سیستم‌عاملOpenSolaris ازتکنیک‌های امنیتی:۱-اجازه دسترسی به فایل‌ها،۲- لیست‌های کنترل دستیابی(ACL)،۳-کنترل دسترسی وظیفه‌ای[1] (RBAC) ۴- تکنیک Privileges پشتیبانی می‌کند. این سیستم‌عامل همچنین از فرمت فایل سیستم‌های متنوعی پشتیبانی می‌کند که مهم‌ترین آنها UFS و ZFS است.

۲-۲-۷- سیستم عامل OpenDarwin

سیستم‌عامل OpenDarwin در سال ۲۰۰۲ توسط شرکت Apple Computer و Internet Systems Consortium شروع شد. این سیستم‌عامل براساس هسته خانواده BSD (بخصوص FreeBSD) و الگوریتم‌های هسته Mach می‌باشد. هدف این سیستم‌عامل به‌وجود آوردن پل ارتباطی بین سیستم‌عامل متن بسته Darwin در شرکت Apple و کمیته‌های متن باز سراسر دنیا می‌باشد. اکثر برنامه‌نویسان این سیستم‌عامل از کارمندان شرکت Apple هستند ولی این کارمندان در توسعه و پیش‌برد این سیستم‌عامل کاملا آزادند و سعی می‌کنند تا در این توسعه با شرکت Apple سازگاری لازم را داشته باشند.

شکل ۲-۷: آرم سیستم عامل Open Drawing

1 Role Based Access Control (RBAC)

سیستم‌عامل OpenDarwin بخاطر تشابه فراوان با سیستم‌عامل‌های خانواده BSD در بین دوست داران و برنامه‌سازان این خانواده مورد توجه است. این سیستم‌عامل با لیسانس کد منبع APSL توزیع می‌شود و شرکت Apple نسخه تجاری این سیستم‌عامل را به‌عنوان سیستم‌عامل سرور (Mac OS X) به فروش می‌رساند. پروژه‌های متن بازی مانند GNU Darwin براساس هسته این‌سیستم‌عامل به وجود آمدند که سعی در کاربرد هسته این سیستم‌عامل در کنار ابزارهای پروژه GNU دارند.

در کل بجز شرکت Apple هیچ شرکت بزرگی فعالیت تجاری گسترده ای براساس این سیستم‌عامل ندارد و لذا OpenDarwin به‌عنوان یک سیستم‌عامل Enterprise در لیست سیستم‌عامل‌های انتخابی این گزارش نیست، ولی به دلایل اهمیت و کاربرد این سیستم‌عامل، در این قسمت ذکر شده است.

۲-۲-۸- سیستم عامل Plan 9

سیستم‌عامل Plan 9 جایگزین سیستم‌عامل یونیکس در Bell Labs خالق یونیکس و Plan 9 بود. سیستم‌عامل Plan 9 محور بسیاری از تحقیقات در Bell Labs بوده است و بهبود فراوانی را نسبت به یونیکس دارد. این سیستم‌عامل دارای محیط چند کاربری توزیع[1]شده است ودر کل یک سیستم عامل توزیع‌شده محسوب می‌شود.

شکل ۲-۸: آرم سیستم عامل Plan 9

سیستم‌عامل Plan 9 به‌عنوان یک پروژه داخلی Bell Labs از اواسط دهه ۱۹۸۰ شروع شده‌است و در سال ۱۹۹۲ اولین نسخه عمومی آن مطابق عرف Bell Labs به دانشگاه‌ها ارائه

1 Distributed Multi-user Environment

شده است. در سال ۱۹۹۵ نسخه دوم این سیستم‌عامل به‌عنوان یک سیستم‌عامل تجاری عرضه و مطرح شد. در اواخر دهه ۱۹۹۰ شرکت Lucent که صاحب مالکیت Bell Labs شده بود، فعالیت و فروش تجاری این سیستم‌عامل را متوقف ساخت. در سال ۲۰۰۰ نسخه سوم این سیستم‌عامل تحت لیسانس نرم‌افزارهای متن باز ارائه شد و در انتها در سال ۲۰۰۲ نسخه چهارم این سیستم‌عامل تحت مجوز جدیدی از نرم‌افزارهای متن باز ارائه شد.

این پروژه همچنان توسط Bell Labs و MIT در حال توسعه است. از ویژگی‌های بارز این سیستم عامل وجود UTF-8 به‌عنوان استاندارد کاراکترهای پایه‌ای سیستم و برنامه‌ها و محیط گرافیکی ویژه و مخصوص به خود است.

سیستم‌عامل Plan 9 از چندین فایل سیستم رایج و فایل سیستم خاص خود به‌نام 9P2000 پشتیبانی می‌کند. این فایل سیستم انحصاری به سیستم‌عامل Plan9 امکانات ویژه توزیع (Distribute) مخصوص به‌خود را می‌دهد.

اگرچه این سیستم‌عامل به لحاظ علمی سرآمد نسخه‌های Unix فعلی است ولی فعالیت تجاری چندانی ندارد و تنها توسط معدودی از شرکت‌ها ارائه و پشتیبانی می‌شود. در نتیجه در مقطع کنونی نمی‌تواند به‌عنوان یک سیستم‌عامل Enterprise و پرکاربرد مطرح شود. در عوض این سیستم‌عامل می‌تواند در دانشگاه‌ها و مراکز تحقیقاتی ایران به‌عنوان یک سیستم‌عامل نوین و سیستم‌عاملی با تکنیک‌های جدید، مطرح شود و به‌عنوان یک الگو برای سیستم‌عامل‌های نسل جدید استفاده شود.

۲-۲-۹- سیستم عامل Inferno

سیستم‌عامل Inferno در اصل در Bell Labs با انشقاق از سیستم‌عامل Plan 9 شروع شده و در حال حاضر توسط Vita Nuova عرضه می‌شود.

این سیستم‌عامل هم می‌تواند مانند یک برنامه برروی یک سیستم‌عامل دیگر اجرا شود و هم می‌تواند مانند یک سیستم‌عامل معمول، بطور مستقیم با سخت‌افزار کامپیوتر در ارتباط باشد. از ویژگی‌های این سیستم‌عامل کد کوچک آن برای هسته سیستم‌عامل است که این امکان را می‌دهد تا از این سیستم‌عامل در کاربرد سیستم‌های تعبیه شده نیز استفاده شود. این سیستم عامل به‌خوبی از خاصیت‌های Grid و Distributed پشتیبانی می‌کند.

برنامه‌های سیستم‌عامل Inferno توسط زبان برنامه‌سازی Limbo نگارش می‌شوند.کد کامپایل شده توسط این زبان در تمام بسترها و سخت‌افزارهای تحت پشتیبانی این سیستم‌عامل اجرا می‌شود. ساختار کد کامپایل شده این زبان در اصل مشابه کد میانی زبان جاوا است که فقط در ماشین مجازی جاوا قابل اجرا است.

در حال حاضر شرکت Vita Nuova این سیستم‌عامل را برای کاربردهای Embedded، Distributed و Grid معرفی می‌کند و این سیستم‌عامل را هم به صورت رایگان و متن‌باز با رعایت مجموعه‌ای از لیسانس‌های مرتبط و هم به صورت خصوصی بدون لیسانس‌های متعدد و در قبال اخذ هزینه عرضه می‌کند.

اهمیت سیستم‌عامل Inferno و زبان برنامه‌سازی Limbo در اهمیت مولد اولیه آنها Bell Labs خالق سیستم‌عامل Unix و زبان برنامه‌سازی C است. این سیستم‌عامل ظاهرا آخرین سیستم‌عاملی بوده که Bell Labs ارائه کرده است.

سیستم‌عامل Inferno پایداری و امنیت لازم را برای مصارف Enterprise دارد، مشروط بر اینکه برنامه‌های لازم برای این سیستم‌عامل نگارش شوند. به علت ساختار API داخلی و خصوصی این سیستم‌عامل و عدم تطبیق این API با POSIX و عدم پشتیبانی این سیستم‌عامل از زبان‌های برنامه‌سازی متداول مانند C و C++ ، امکان استفاده از این سیستم‌عامل در مصارف معمول وجود ندارد و فقط می‌توان از آن در پروژه‌های خاص استفاده کرد.

۲-۲-۱۰- سیستم عامل ReactOS

سیستم‌عامل ReactOS سیستم‌عامل متن‌بازی است که اگرچه جزء سیستم‌عامل‌های کاربردی و عملیاتی نیست، ولی به دلایل اهمیت آن در اینجا اشاره شده‌است. سیستم‌عامل ReactOS پروژه‌ای است که سعی دارد تا سیستم‌عامل ویندوز NT محصول شرکت مایکروسافت را به صورت متن باز پیاده‌سازی کند. اگر چه این سیستم‌عامل هنوز در مراحل ابتدای خود به سر می‌برد و هنوز کامل نیست، ولی قادر است تعدادی از برنامه‌های سیستم‌عامل ویندوز را به خوبی اجرا کند. پروژه این سیستم‌عامل در ابتدا در سال ۱۹۹۶ با هدف پیاده‌سازی ویندوز ۹۵ شروع شد، ولی تا سال ۱۹۹۷ فعالیت چندانی نداشت و در سال ۱۹۹۸ فعالیت جدی خود را بر روی هسته این سیستم‌عامل آغاز کرد.

شکل ۲-۹: آرم سیستم عامل ReactOS

یکی از مشکلات دست و پاگیر این پروژه، اتهاماتی است که به تیم توسعه این سیستم‌عامل و کد منبع آن وارد شده‌است و در این اتهامات گفته می‌شد که قسمتی از کد این سیستم‌عامل همان کد ویندوز شرکت مایکروسافت است که مهندسی معکوس شده است. در حال حاضر یک تیم نظارتی در داخل این پروژه تشکیل شده تا این اطمینان حاصل شود که هیچ کجای کد منبع این پروژه از نسخه مهندسی معکوس شده کد ویندوز شرکت مایکروسافت نباشد و از طرف دیگر بین برنامه‌سازان و تیم توسعه این سیستم‌عامل قراردادهایی برای اطمینان بیشتر وضع شده است.

رشد این سیستم‌عامل و تکمیل شدن آن بخصوص برای ما ایرانیان که به ویندوز بسیار عادت کرده‌ایم بسیار سودمند است. در صورتیکه این سیستم‌عامل بتواند به تدریج کامل‌تر شود تا جایی که در محیط‌های عملیاتی بتوان از آن استفاده کرد، می‌توان این سیستم‌عامل را به سادگی در بسیاری از کاربردها، جایگزین ویندوز کرد. این سیستم‌عامل قصد دارد تا هم در لایه ارتباط با برنامه‌ها (API) و هم در لایه راه‌انداز سخت‌افزارها[1] با ویندوز هماهنگی لازم را داشته باشد.

۲-۲-۱۱- سیستم عامل FreeDOS

سیستم‌عامل FreeDOSسیستم‌عامل متن‌بازی است‌که به‌عنوان یک‌جایگزین برایMS-DOS مطرح است و ۱۰۰٪ منطبق باMS-DOS است. اگرچه سیستم‌عامل MS-DOS و جایگزین‌های

1 Device Drivers

این سیستم‌عامل قطعا نمی‌تواند به‌عنوان سیستم‌عامل‌های Enterprise مطرح باشند، ولی این سیستم‌عامل‌ها می‌توانند چاره بسیاری از مشکلات ما باشند. سیستم‌عامل FreeDOS در حال حاضر به‌طور وسیعی در کاربردهای صنعتی و سیستم‌های تعبیه شده و بلادرنگ استفاده می‌شود. سادگی سیستم‌عامل MS-DOS و وجود ابزارهای متعدد، شناخته شده و ارزان‌قیمت برای توسعه و برنامه نویسی از یک‌سو و سبک بودن این سیستم‌عامل از سوی دیگر، آن را به یک سیستم‌عامل ایده‌آل در کاربردهای خاص و بخصوص در صنعت کرده است.

شکل ۲-۱۰: آرم سیستم عامل FreeDOS

سیستم‌عامل FreeDOS بعضی اشکالات نرم‌افزاری MS-DOS مانند مدیریت حافظه را رفع کرده‌است و با وجود این کاملا با سیستم‌عامل MS-DOS سازگار است. هر روز به شمار برنامه‌هایی که در سیستم‌عامل FreeDOS قابل استفاده هستند، اضافه می‌شود و زبان‌هایی مانند FeePASCAL برای این سیستم‌عامل وجود دارند که برای تدریس در دانشگاه‌ها بسیار مناسب هستند.

وجود چنین سیستم‌عاملی بخصوص برای دانشجویان ایرانی و اساتید دانشگاه‌های ایران که به ابزارهای کهنه مانند Turbo C++ و Turbo Pascal عادت کرده‌اند، روزنه امیدی به راه‌یابی به مباحث و سیستم عامل‌های مطرح امروزی دنیای صنعت است.

۲-۳- مقایسه عمومی انواع سیستم عامل‌های متن باز

در جدول(۲-۱) مقایسه عمومی انواع سیستم‌عامل‌های متن باز شامل شرکت یا گروه تولید کننده، اولین تاریخ انتشار عمومی، پروژه مادر، نوع لیسانس مادر و نوع کامپیوترهایی که معمولا از این سیستم‌عامل استفاده می‌کنند وجود دارد.

جدول۲-۱: مقایسه عمومی انواع سیستم عامل‌های متن باز

سیستم عامل	خالق	اولین نسخه	پروژه مادر	لیسانس غالب	نوع استفاده
FreeBSD	The FreeBSD Project	December 1993	386BSD	BSD	Server, Workstation, Network Appliance
Inferno	Bell Labs	1997	Plan 9	MIT/GPL/LGPL/LPL	Network Appliance, Server, Embedded
GNU/ Linux	Various authors	September 17, 1991	Minix	Usually GNU GPL/Copy left	بسته به توزیع کننده
NetBSD	The NetBSD Project	May 1993	386BSD	BSD	Network Appliance, Server, Workstation, Embedded
OpenBSD	The OpenBSD Project	October 1995	NetBSD 1.0	BSD	Server, Network Appliance, Workstation, Embedded
Plan 9	Bell Labs	1993	Unix	LPL	Workstation, Server, Embedded, HPC
Solaris	Sun	July 1992	SunOS	CDDL	Server, Workstation

. جدول ۲-۲: مقایسه تکنیکی انواع سیستم عامل‌های متن باز

سیستم عامل	انواع سخت‌افزار	انواع فایل سیستم	نوع هسته	محیط گرافیکی	جامعه‌ی پیرامونی	مدیریت به‌روزرسانی	API اصلی
FreeBSD	x86, AMD64, PC98, SPARC, others	UFS2, ext2, FAT, ISO 9660, UDF, NFS, ReiserFS (read only), XFS (experimental) and others	Monolithic with modules	No	ports tree, packages	by source (CVSup, portsnap), network binary update (freebsdupdate)	BSD, POSIX
Linux	x86, Alpha, AMD64, PPC, SPARC, others	ext2, ext3, ReiserFS, FAT, ISO 9660, UDF, NFS, and others	Monolithic with modules	No	بستهٔ به توزیع-کننده	POSIX	--
Inferno	x86, Alpha, MIPS, PPC, SPARC, others	Styx/9P2000, kfs, FAT, ISO 9660	Monolithic with modules, user space file systems	Yes	--	--	خصوصی
NetBSD	x86, 68k, Alpha, AMD64, PPC, SPARC, many others	UFS, UFS2, ext2, FAT, ISO 9660, NFS, LFS, and others	Monolithic with modules	No	pkgsrc	by source (CVS, CVSup, rsync) or binary (using sysinst)	BSD, POSIX
OpenBSD	x86, 68k, Alpha, AMD64, SPARC, VAX, others	ffs, ext2, FAT, ISO 9660, NFS, some others	Monolithic with modules	No	ports tree, packages	by source	BSD, POSIX
Plan 9	x86, Alpha, MIPS, PPC, SPARC, others	fossil/venti, 9P2000, kfs, ext2, FAT, ISO 9660	Monolithic, user space file systems	Yes	None	replica	Unix-like (and optional POSIX compatibility layer)
Solaris	SPARC, SPARC64, AMD64, x86	UFS, ZFS, ext2, FAT, ISO 9660, UDF, NFS, some others	Monolithic with modules	Yes	SysV packages (pkgadd)	Sun Update Connection	SysV, POSIX

جدول ۲-۳: مقایسه امکانات امنیتی انواع سیستم عامل‌های متن باز

سیستم عامل	کنترل منابع	محیط ایزوله	دیوار آتش	Encrypted file system	Data execution prevention		اشکالات امنیتی شناخته شده	
					Hardware	Emulation	Number	Oldest
FreeBSD	Unix, ACLs, MAC	chroot, jail, MAC Partitions	IPFW2, IPFilter, PF	Yes	n/a		0	-
Inferno	Unix	Namespaces	n/a	n/a	No	No	n/a	
Linux	Unix, ACLs, MAC	chroot, capabilities, seccomp, SELinux	Netfilter متغیر بسته به توزیع	Yes	Yes	No	متغیر بسته به توزیع	
NetBSD	Unix, Veriexec	chroot, systrace	IPFilter, PF	Yes	Yes	No	n/a	
Open BSD	Unix	chroot, systrace	PF	Yes	Yes	Yes	0	-
Plan 9	Unix	Namespaces	ipmux	Yes	No	No	n/a	
Solaris	Unix, Role Based Access Control (RBAC)	Chroot, Zones	IPFilter	n/a	Yes	No	2	2005-04-13

۲-۴- مقایسه تکنیکی انواع سیستم عامل‌های متن باز

در جدول (۲-۲) سیستم‌عامل‌های متن باز به لحاظ امکانات فنی بررسی شده‌اند. موارد بررسی شده عبارتند از: ۱- انواع سخت‌افزار حمایت‌شده توسط سیستم‌عامل، ۲- انواع سیستم فایل‌های قابل پشتیبانی در سیستم‌عامل، ۳- نوع هسته سیستم‌عامل، ۴- وجود محیط گرافیکی پیش‌فرض در سیستم‌عامل، ۵- سیستم مدیریت بسته‌ها و برنامه‌ها، ۶- نحوه به

روزرسانی سیستم‌عامل توسط خالق اولیه، ۷-نحوه محاوره سیستم‌عامل با برنامه‌ها (API اصلی) .

طبق جدول مذکورسیستم‌عامل‌های لینوکس خانواده BSD و OpenSolaris از بیشترین امکانات تکنیکی برخوردار هستند.

۲-۵- مقایسه امکانات امنیتی انواع سیستم عامل‌های متن باز

در جدول(۲-۳) مقایسه امکانات‌امنیتی انواع سیستم‌عامل‌های متن‌باز شامل:۱-انواع تکنیک‌های کنترل منابع،۲- انواع تکنیک‌های تولید محیط ایزوله برای برنامه‌ها،۳- وجود دیواره آتش پیش فرض برای سیستم‌عامل،۴- امکان رمزنگاری سیستم فایل‌ها و اطلاعات،۵- مقابله با اجرای داده‌ها به جای کد برنامه، ۶-تعداد اشکالات امنیتی شناخته شده و رفع نشده طبق جدول فوق، سیستم‌عامل‌های Linux، FreeBSD و OpenSolaris از بیشترین امکانات امنیتی پشتیبانی می‌کنند. البته دقت شود که پشتیبانی از امکانات امنیتی بیشتر به معنای امن‌تر بودن سیستم‌عامل به صورت ذاتی نیست و مدیران ارشد سیستم‌عامل‌ها از این امکانات امنیتی باید به درستی استفاده کنند و آنها را با نیازهای خود منطبق کنند تا نتیجه مطلوب‌تری حاصل شود.

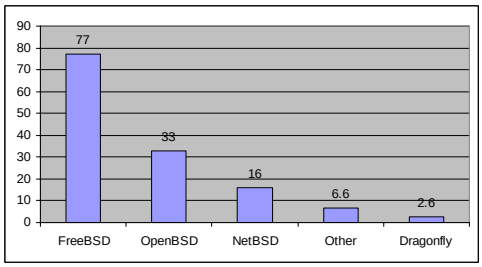
۲-۶- مقایسه اعضای سیستم عامل‌های خانواده BSD

مقایسه کردن خانواده BSD شامل FreeBSD، NetBSD، OpenBSD، Dragonfly و بسیاری اعضای دیگر این خانواده به سادگی امکان‌پذیر نیست، زیرا این سیستم‌عامل‌ها رقابت زیادی با یکدیگر دارند و به علت یکسان بودن لیسانس کد منبع، این سیستم‌عامل‌ها به سرعت شبیه به یکدیگر می‌شوند.

حتی این مساله که کدامیک از این سیستم‌عامل‌ها طرفداران بیشتری دارند، چندان روشن نیست. در سال ۲۰۰۵ موسسه BSD Certification Group، با یک اعلان همگانی در چندین وب سایت و لیست‌های پست الکترونیکی از کاربران سیستم‌عامل‌های خانواده BSD دعوت کرد تا در یک سرشماری شرکت کنند و نتیجه این سرشماری این بود که بطور تقریبی، ۷۷٪ کاربران از FreeBSD استفاده می‌کنند، ۳۳٪ از OpenBSD استفاده می‌کنند، ۱۶٪ از NetBSD استفاده می‌کنند، ۲.۶٪ از Dragonfly استفاده می‌کنند و ۶.۶٪ از باقی این خانواده

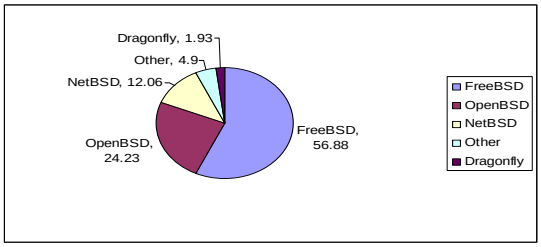
استفاده می‌کنند. دقت شود که در این سرشماری کاربران می‌توانستند بیش از یک گزینه را انتخاب کنند و لذا جمع اعداد فوق بیشتر از ۱۰۰٪ است.

نمودار شکل(۲-۱۱) که از اعداد فوق استخراج می‌شود، نشان می‌دهد که عمده کاربران خانواده BSD، با سه سیستم‌عامل FreeBSD، OpenBSD و NetBSD کار می‌کنند و دیگر سیستم‌عامل‌های این خانواده چندان محبوب و پراستفاده نیستند.



شکل ۲-۱۱: آمار مصرف

در صورتیکه از اعداد فوق درصد گرفته شود نمودارشکل (۲-۱۲) حاصل می‌شود.



شکل ۲-۱۲: درصد مصرف خانواده BSD

نمودار فوق مجددا به اهمیت سیستم‌عامل FreeBSD در میان سه عضو دیگر این خانواده اشاره می‌کند. اهمیت سیستم‌عامل FreeBSD در تیم خالق FreeBSD نهفته است. همانطور که گفته شد، سایت‌های Yahoo!، Hotmail و Cdrom.com هم از همین سیستم‌عامل استفاده می‌کنند.

از جمله دلایل محبوبیت سیستم‌عامل FreeBSD پشتیبانی این سیستم‌عامل از بسیاری از سخت‌افزارهای مختلف در کامپیوترهای شخصی است (برخلاف OpenBSD و NetBSD). کلا این سیستم‌عامل بخصوص برای کامپیوترهای شخصی x86 بسیار بهینه شده‌است. این

سیستم‌عامل نصب ساده‌ای نسبت به سایر اعضای این خانواده دارد و پشتیبانی و بروز رسانی قوی‌تری دارد.

در چند ماه گذشته، FreeBSD لیسانس توزیع Java را نیز از شرکت Sun دریافت کرد که سبب افزایش محبوبیت بیش از پیش این سیستم‌عامل نیز خواهد شد.

۲-۷- توزیع‌های مختلف لینوکس

سیستم‌عامل لینوکس بر عکس سیستم‌عامل FreeBSD و خانواده آن، که توسط یک گروه تولید و توزیع می‌شود، توسط گروه‌های متعدد تولید می‌شود و شرکت‌ها و سازمان‌های متعدد این سیستم‌عامل را توزیع می‌کنند. برای مثال هسته سیستم‌عامل لینوکس به سرپرستی آقای لینوس توروالدس توسعه می‌یابد، در حالیکه عمده عناصر سیستم‌عامل لینوکس یا در پروژه GNU توسعه پیدا می‌کنند یا توسط افراد و گروه‌های مستقل مانند Apache Group که معروف‌ترین سرور وب دنیا را نگارش کرده است، توسعه پیدا می‌کنند.

در این میان شرکت‌هایی مانند Red Hat و Novel و گروه‌هایی مانند Debian و Slackware نتیجه کار گروه‌های متعدد را جمع آوری می‌کنند و در قالب بسته‌های متعدد مانند CD و DVD توزیع‌می‌کنند. بعضی از این شرکت‌ها مانند Red Hat در کنار بسته‌بندی خود سرویس‌های پشتیبانی و بروزرسانی را نیز انجام می‌دهند و در قبال خدمات خود، از مشتریان هزینه اخذ می‌کند.

وجود چندین توزیع‌کننده سبب اختلافات فراوان نرم‌افزاری میان توزیع‌های مختلف لینوکس شده است و کار را برای شرکت‌هایی که برای لینوکس برنامه می‌نویسند، مشکل ساخته است. توزیع کنندگان لینوکس با وضع استانداردهای مختلف که به استانداردهای پایه‌ای لینوکس معروف است، سعی می‌کنند به لحاظ نرم‌افزاری خود را یکسان و شبیه به یکدیگر نگه دارند تا سایر شرکت‌ها بتوانند به سادگی برای این توزیع‌ها برنامه‌سازی کنند.

در جدول (۲-۴) لیست بسیار کوچکی از معروف‌ترین توزیع‌کنندگان لینوکس وجود دارد که بالغبر ۵۵ عنوان است. لیست نسبتا کاملی ازتوزیع‌کنندگان درسایت http://www.linux.org/dist موجود است که چند صد توزیع‌کننده لینوکس در آن ثبت شده‌اند.

جدول ۲ - ۴: توزیع‌های مختلف لینوکس

Name	Creator	Producer	Predecessor	Cost	License	Target audience	Origin
aLinux	Jay Klepacs		Red Hat Linux	Free	GPL	Desktop, Workstation, Server, Windows users	Canada
ALT Linux	ALT Linux Team		Mandrake Linux	Free	GPL	Desktop, Workstation, Server, Enthusiast	Russia
Annvix	Vincent Danen		Mandrake Linux	Free	GPL	Server	Canada
Arch Linux	Judd Vinet	dev team	CRUX	Free	GPL	Workstation, Server, Enthusiast	Canada
Ark Linux	Bernhard Rosenkraenzer	dev team	None	Free	GPL	Beginners, Desktop, Workstation, Windows users	Switzerland
Arudius	Haidut	dev team	None	Free	GPL	Information assurance, information security users	USA
Asianux	Miracle Linux, Red Flag Linux, Haansoft Linux	dev team	RHEL	Free	GPL	Server, good for Desktop	East Asia
Aurox	Software-Wydawnictwo Sp. z oo	Aurox Sp. z oo	Red Hat Linux	Free	GPL	Desktop, Workstation, Server, Enthusiast	Poland
Caixa Mágica	Daniel Neves, José Guimarães, Paulo Trezentos	Caixa Mágica	None	Free (download editions)		Desktop, Server (not free)	Portugal
CentOS	CentOS Project	CentOS Project	RHEL	Free	GPL	Desktop, Workstation, Server, Enthusiast	USA
CRUX	Per Liden	CRUX Linux community	None	Free	GPL	Desktop, Workstation, Server, Enthusiast	Sweden
Damn Small Linux	John Andrews	dev team	Knoppix	Free	GPL	Workstation, Server, Enthusiast	USA
DARKSTAR Linux	DARKSTAR Linux Project	DARKSTAR Linux Project	Slackware	Free	GPL	Workstation, Server, Enthusiast	Romania

Name	Creator	Producer	Predecessor	Cost	License	Target audience	Origin
Debian	Ian Murdock	Debian Project	none	Free	any DFSG-free	Desktop, Workstation, Server, Enthusiast	USA
DeLi Linux	Henry Jensen		None	Free	GPL	Desktop	Germany
DeMuDi	dev team	AGNULA	Debian	Free	GPL	DAW, Desktop, Workstation	Europe
dyne:bolic	Jaromil		none	Free	GPL	Windows users, DAW, Desktop, Workstation	Italy
Fedora Core	Fedora Project		Red Hat Linux	Free	GPL	Desktop, Workstation, Server, Enthusiast	USA
Finnix	Ryan Finnie		Debian	Free	GPL	Administration, Server	USA
Fox Linux	Dev team, supported by ILDN network		Fedora	Free	GPL	Desktop	Italy
Frugalware	Miklos Vajna	dev team	Slackware	Free	GPL	Workstation, Server, Enthusiast	Hungary
Gentoo Linux	Daniel Robbins	Gentoo Foundation, Inc.	none	Free	GPL	Desktop, Workstation, Server, Enthusiast	USA
Gnoppix	Klaus Knopper	dev team	Debian, Knoppix & Ubuntu	Free	GPL	Desktop, Workstation, Windows users	Germany
GoboLinux	Hisham Muhammad, Andre Detsch	dev team	none	Free	GPL	Workstation, Server, Enthusiast	Brazil
ImpiLinux	Gauteng Linux Users Group	ImpiLinux (Pty) Ltd.	Debian, Gnoppix, Knoppix & Ubuntu	Free	GPL	Workstation, Server, Enthusiast	South Africa
Kanotix	Kano	dev team	Knoppix	Free	GPL	Desktop, Workstation, Windows users	Germany
Knoppix	Klaus Knopper	dev team	Debian	Free	GPL	Desktop, Workstation, Enthusiast, Windows users	Germany
Kurumin Linux	Carlos Morimoto	Guiado Hardware	Knoppix	Free	GPL	Desktop	Brazil
Linspire	Lindows. com, Inc.	Linspire, Inc	Debian	49.95 $	GPL/Commercial	Beginners, Desktop, Windows Users	USA
Lunar Linux	Chuck Mead, Lunar	dev team	Sorcerer	Free	OSI Approved	Workstation, Server, Enthusiast	USA

Name	Creator	Producer	Predecessor	Cost	License	Target audience	Origin
	Penguin Project						
Lycoris Desktop/ LX	Redmond Linux Corp.	Lycoris, Inc., Mandriva S.A.	Caldera OpenLinux	40-75$	Lycoris License	Workstation	USA
Mandriva Linux	Mandrakesoft S.A.	Mandriva S.A.	Red Hat Linux	Free	GPL	Beginners, Desktop, Workstation, Server	France, Brazil
MeNTOPPIX	Galuh Prasetyawan	Galuh Prasetyawan	Debian, Knoppix	Free	GPL	Desktop, Windows users	Indonesia
MEPIS	Warren Woodford	MEPIS LLC.	Debian	Free	GPL	Beginners, Desktop, Windows Users	USA
NimbleX	Bogdan Radulescu		Slackware	Free	GPL	Desktop, Workstation, Newbie, Enthusiast, Windows Users	Romania
PCLinuxOS	Texstar	dev team	Mandriva	Free	GPL	Desktop, Windows Users	Texas, USA
PLD Linux	PLD Linux Distribution		none	Free	GPL	Workstation, Server, Enthusiast	Poland
Puppy Linux	Barry Kauler	Puppy Foundation	none	Free	GPL/LGPL	Beginners, Desktop, Embedded, Enthusiast	Australia
Red Flag Linux	Institute of Software, Chinese Academy of Sciences, NewMargin Venture Capital	Red Flag Software Co., Ltd.	Red Hat Linux	Free,	GPL	Workstation, Server, Enthusiast	China
Red Hat Linux	Red Hat	replaced with Fedora Core	n/a	$40, Obsolete	GPL	Desktop, Workstation, Server, Enthusiast	USA
RHEL	Red Hat & Fedora Project		Red Hat Linux	$180-2,500	GPL	Workstation, Server, Businesses	USA
Rxart	Pixart SRL	Pixart SRL	Debian	$14.95	GPL Commercial	Desktop, Workstation, Server, Enthusiast, Windows users	Argentina
Scientific Linux	CERN, Fermilab, dev team		Fermi (LTS) & RHL/RHEL	Free	GPL	Desktop, Workstation, Server	Switzerland, USA

Name	Creator	Producer	Predecessor	Cost	License	Target audience	Origin
Slackware	Patrick Volkerding	dev team	SLS	Free	GPL	Workstation, Server, Enthusiast	USA
Source Mage GNU/Linux	Ryan Abrams, Eric Schabell	dev team	Sorcerer	Free	GPL	Workstation, Server, Enthusiast	N/A
openSUSE	Novell, Inc and OpenSource community		Jurix	Free	GPL	Desktop, Workstation, Server, Enthusiast	Germany
Symphony OS	Ryan Quinn	Ryan Quinn & Jason Spisak	Debian	Free	GPL Version 2	Desktop, Workstation	N/A
Trustix Secure Linux	Comodo Group Inc.		Red Hat Linux	Free	GPL	Server	England
Ubuntu Kubuntu Edubuntu Xubuntu	Canonical Ltd.		Debian	Free	GPL	Beginners, Desktop, Workstation, Enthusiast, Server	Isle of Man
Ultima Linux	Martin Ultima		Slackware	Free	GPL	Desktop, Workstation, Enthusiast, Server	USA
UTUTO	Diego Saravia & Daniel Olivera	UTUTO dev-team	Gentoo	Free	GPL-2 or later	Freedom lovers, Desktop, Workstation, Server, Enthusiast	Argentina
Vector Linux	Robert S. Lange	dev team	Slackware	Free	GPL	Desktop, Workstation, Enthusiast	Canada
Xandros Desktop OS	Xandros Corp.	Xandros Inc.	Corel	$50-$90	Commercial/ GPL	Beginners, Windows Users, Desktop, Workstation, Server	Canada
Zenwalk	Jean-Philippe Guillemin	dev team	Slackware	Free	GPL	Desktop, Workstation, Enthusiast	(unknown)

با توجه به لیست فوق، مشاهده می‌شود که منشا عمده لینوکس‌ها بر پایه Red Hat، Red Hat Enterprise Linux (RHEL)، Debian و Slackware است که این به سبب قدمت و یا اهمیت این توزیع‌ها است. یکی از توزیع‌های بسیار معروف دیگر در اروپا SUSE است که در

سال‌های اخیر شرکت آمریکایی Novel آن را خریده است و در ایران برخی متخصصین عمدتا بخاطر سادگی، زیبایی و کارایی این توزیع‌کننده به آن علاقمند هستند.

در صورتیکه در یک پروژه ایرانی لازم است از یک لینوکس خارجی استفاده شود، بهتر است از بین توزیع‌های معروف که امکان بروزرسانی آنها در ایران وجود دارد مانند CentOS، OpenSUSE، Debian و Fedora استفاده شود. توزیع‌کننده CentOS بخصوص پیشنهاد مناسبی برای بسیاری از پروژه‌ها و نیازها ست. این توزیع‌کننده نسخه رایگان و متشابه با Red Hat Enterprise Linux است و با بسیاری از برنامه‌های تجاری و غیرتجاری مانند Oracle به خوبی سازگار است. این توزیع‌کننده به خوبی در ایران قابل دستیابی و بروزرسانی است و خوشبختانه به سبب شباهت نظیربه‌نظیر آن با RHEL از لحاظ مستندات فنی بسیار غنی است. این توزیع‌کننده هم‌اکنون در ۴ لوح فشرده قابل نصب و ۴ لوح فشرده کد منبع توزیع‌می‌شود و شامل چهار گزینه نصب: ۱-کامپیوترهای شخصی،۲ -کامپیوترهای کاری، ۳-کامپیوتر سرور، ۴- نصب دلخواه است. این توزیع‌کننده همچنین توسط برنامه‌های Yum و Up2date به خوبی به‌روز می‌شود.

۲-۸- مقایسه کلی لیسانسGPL و BSD

اگرچه لیسانس BSD و GPL هر دو لیسانس‌های معمول در جامعه متن باز هستند ولی این دو لیسانس اختلافاتی در نوع برخورد با کد منبع اولیه دارند. لیسانس GPL با هدف متن باز بودن همیشگی کد منبع طراحی شده است. هر فرد یا گروهی که یک برنامه با لیسانس GPL را توزیع‌کند حتی اگر کد منبع اولیه را تغییر داده باشد، باید کد منبع نهایی را به همراه برنامه خود توزیع‌کند. این ویژگی باعث می‌شود تا کد منبع GPL در هیچ برنامه متن بسته‌ای ظاهر نشود.

لیسانسBSD اینگونه نیست. در این لیسانس کد منبع برنامه اولیه را می‌توان با آزادی کامل تغییر داد و در یک برنامه متن بسته توزیع‌کرد، مشروط براینکه موارد قید شده در لیسانس BSD همچنان وجود داشته باشد و توزیع‌کننده ذکر کرده‌باشد که از کد برنامه با مجوز BSD استفاده کرده است. در حال حاضر کدIPStack درWindows و Mac OS X از کدی با لیسانس BSD منشق شده‌است و در پروژه‌های متن بسته ذکر شده، استفاده شده‌است.

هسته سیستم‌عامل لینوکس دارای لیسانس GPL است، پس هیچگاه این هسته نمی‌تواند به صورت متن بسته توزیع‌شود و سیستم‌عامل‌های خانواده BSD عمدتا از لیسانس BSD استفاده می‌کنند و می‌توان از کد آنها در هر پروژه‌ای استفاده کرد.

۲-۹- نتایج

در اینجا سعی شده از بین سیستم‌عامل‌های متن باز بررسی شده، یکی انتخاب شود. توجه کنید که سیستم‌عامل‌های عنوان‌شده در این بخش، گرچه همگی متن‌باز هستند و مشخصات کلی آنها و نقاط ضعف و قوت آنها کاملا مشخص است، ولی نمی‌توان بطور قطعی از بین همه این سیستم‌عامل‌ها یک سیستم‌عامل و فقط یک سیستم‌عامل را جوابگوی تمام نیازها دانست و آن را به‌عنوان بهترین سیستم‌عامل تصور کرد. اگر اینگونه بود در تمام دنیا پروژه توسعه بقیه سیستم‌عامل‌ها متوقف می‌شد و همه دنیا فقط از یک سیستم‌عامل استفاده می‌کرد.

به‌طور قطع سیستم‌عامل Linux، FreeBSD و OpenSolaris نقش پررنگ‌تری در سیستم‌عامل‌های Enterprise دارند. بخصوص که بسیاری از شرکت‌های بزرگ که عنوان Enterprise را در محصولات خود دارند فقط نسخه‌های ویژه سیستم‌عامل Linux و Solaris را ارائه می کنند و چاره ای برای مشتریان خود قرار نمی‌دهند.

سیستم‌عامل FreeBSD از قدیم بخصوص در نقاطی که سیستم‌عامل ناچار به تحمل بار سنگین شبکه است، مانند Cache و Ftp Server معروف بوده است. سیستم‌عامل FreeBSD و کلا خانواده BSD خصوصا بخاطر نوع مجوزی که دارند، اجازه می‌دهند تا در محصولات خصوصی که بنابه دلایل مختلف نمی‌توان کد منبع آنها را ارائه کرد، استفاده شوند.

سیستم‌عامل OpenSolaris محصول شرکت Sun شاید جزء بی‌نظیرترین و کامل‌ترین سیستم‌عامل‌ها باشد. این سیستم‌عامل با مجموعه وسیعی از ابزارهای برنامه‌سازی، ابزارهای مختلف شبکه و بسیاری از ابزارهای دیگر، محیط بسیار کاملی را هم به مدیران ارشد سیستم، هم به برنامه‌سازان و هم به کاربران عادی سیستم‌عامل ارائه می‌کند. مشکل بزرگ این سیستم‌عامل مخفی بودن قسمت کوچکی از کد این سیستم‌عامل و بسته بودن سایت این سیستم‌عامل برای ایرانیان است. شرکت Sun اجازه دریافت این سیستم‌عامل و بسیاری از محصولات دیگر خود را به کاربرانی که از محدوده IPهای ایران به این سایت متصل می‌شوند، نمی‌دهد.

با توجه به اینکه در کشورما عمدتا مدیران ارشد شبکه عادت به سیستم‌عاملWindows دارند و خوشبختانه با تبلیغات و حرکت‌های داخلی ارگان‌های مختلف، سیستم‌عامل لینوکس به‌عنوان یک سیستم‌عامل قدرتمند و کارا در میان عامه متخصصین شناخته شده‌است، مسلما به دلایل فوق بهترین انتخاب برای سیستم‌عامل Enterprise در حال حاضر در کشور ما Linux است.

البته مدیران ارشد، محققین و برنامه‌سازان هرگز نباید خود را فقط وابسته به لینوکس ببینند و همواره درنظر داشته باشند که متشابهات سیستم‌عامل لینوکس مانند FreeBSD و OpenSolaris همواره وجود دارند و بنابه صلاح‌دید، در یک پروژه می‌توان از این سیستم‌عامل‌ها و سیستم‌عامل‌های دیگر استفاده کرد یا حتی با صرف اندکی وقت، هزینه و استفاده از کدهای منبع معتبر موجود در اینترنت و سیستم‌عامل‌های متن باز رایج، یک سیستم‌عامل خاص کوچک و نوین تولید کرد.

فصل سوم – بررسی و شناسایی تغییرات لازم جهت بومی سازی سیستم عامل لینوکس

در قسمت قبل، سیستم‌عامل لینوکس به عنوان یک سیستم‌عامل Enterprise که جوابگوی بسیاری از کاربردهاست معرفی شد. همانطور که گفته شد، سیستم عامل لینوکس، بر خلاف سیستم‌عامل‌های Windows ، FreeBSD و OpenSolaris که توسط یک مجموعه یا شرکت خاص ارائه می شوند، به صورت نرم‌افزارهای مختلف توسط افراد، گروه‌ها و سازمان‌های مختلف تولید می‌شود و توزیع کنندگان لینوکس، اجزاء دلخواه خود در را کنار یکدیگر قرار می‌دهند و این سیستم‌عامل را در قالب بسته‌های متنوع با کاربردهای گوناگون توزیع می‌کنند. به این ترتیب نمی‌توان یک تعبیر و یا تعریف جامع از سیستم‌عامل لینوکس یا دیسک توزیع کننده این سیستم‌عامل ارائه داد و آن را جزء به جزء بررسی کرد.

مهمترین قسمت سیستم‌عامل لینوکس، هسته این سیستم‌عامل است که آقای لینوس به همراه تیم خود آن را تولید می‌کند. هسته سیستم‌عامل لینوکس در توزیع‌های مختلف، با توجه به نیازهای نرم‌افزاری و سخت‌افزاری، ممکن است تغییر یافته باشد. برای نمونه شرکت Red Hat که یکی از مهمترین توزیع کنندگان لینوکس است، پس از اعمال تغیرات نسبی فراوان، از هسته لینوکس در توزیع معروف RHEL خود استفاده می‌کند و این سبب می‌شود تا حتی هسته لینوکس در توزیع‌های مختلف با یکدیگر اختلاف داشته باشد. با در نظر گرفتن نکات فوق، لینوکس Enterprise برای استفاده در ایران، تعریف از پیش مشخص و کلیشه ای ندارد و از این رو کاملا قابل تعریف است و بومی سازی این سیستم‌عامل می‌تواند در جهت این تعریف باشد.

قابل ذکر است که هدف از سیستم‌عامل بومی Enterprise استفاده از آن در محیط شبکه و یا زیر بار سنگین و مشابهات آن است و استفاده از آن به عنوان یک محیط رومیزی چندان مد نظر نیست.

۳–۱– تعریف نرم‌افزار و سیستم عاملEnterprise

نرم‌افزار Enterprise به نرم‌افزاری گفته می‌شود که یک راه‌کار قابل قبول برای یک مشکل Enterprise ارائه کند. مشکل Enterprise، اشکالی است که توسط نرم‌افزاری‌های معمول و عادی حل نمی‌شود و برای حل آن احتیاج به برنامه‌های Enterprise وجود دارد. برنامه‌های

Enterprise برنامه‌هایی هستند که معمولا با ساختارهای (مدل‌های) Enterprise به کمک ابزارهای Enterprise طراحی و تولید شده و در کنار سایر برنامه‌های Enterprise به کار گرفته می‌شوند.

برنامه‌های Enterprise امکاناتی دارند که معمولا در برنامه‌های دیگر یافت نمی‌شوند، برای مثال بانک‌های اطلاعاتی Enterprise تجاری و متن بسته مانند Oracle و Ms SQL مزایای قابل توجهی نسبت به سایر بانک‌های اطلاعاتی متن بسته دیگر مانند Ms Access دارند. با توجه به تعریف فوق سیستم‌عامل Enterprise نیز می‌باید مزایایی نسبت به سایر سیستم‌عامل‌ها داشته باشد تا بتوان از آن در کاربردهای Enterprise استفاده کرد. با توجه به قسمت‌های قبل، می‌دانیم که سیستم‌عامل لینوکس با هدف تولید نسخه رایگانی از سیستم‌عامل UNIX که یک سیستم‌عامل کاملا Enterprise است، طراحی و تولید شده است. از این رو طبیعتاً سیستم‌عامل Linux با در نظر گرفتن مسائل مطرح دردنیای سیستم‌عامل‌های Enterprise طراحی شده است و با توجه به استقبال روزافزون از این سیستم‌عامل در دنیای Enterprise، این سیستم‌عامل امتحان خود را پس داده و به عنوان یک سیستم‌عامل Enterprise قابل قبول در سطح جهانی است.

در سیستم‌عامل لینوکس نکته‌ای وجود دارد که می‌باید به آن توجه داشت؛ توزیع‌های لینوکس بسیار متنوع و متفاوت هستند و همه این توزیع‌ها، هدف ارائه یک سیستم‌عامل Enterprise را ندارند. بسیاری از نسخه‌های سیستم‌عامل لینوکس برای کاربردهای متنوع دیگر طراحی شده‌اند و مناسب برای راهکارهای Enterprise که معمولا در آنها هدف کاربردهای شبکه‌ای و کاربردهای سنگین دیگر است، نیستند. از سوی دیگر، بعضی از توزیع‌های این سیستم‌عامل مانند Red Hat و Novell در کاربردهای Enterprise بسیار صاحب نام هستند و در تعریف و تدوین سیستم عامل Enterprise بومی، می‌توان از این سیستم‌عامل‌ها الگو برداری مناسب کرد و از مزایای آنها سود جست.

۳-۲- ویژگی‌های مطلوب برای سیستم عامل بومیEnterprise

ازآنجا که بحث ما درارتباط با سیستم‌عامل لینوکس است، ازذکر ویژگی‌های لازم و درونی این سیستم‌عامل Enterprise مانند امنیت، چند پردازندگی، پایداری و غیره صرف نظر می‌کنیم و به ویژگی‌هایی می‌پردازیم که برای بومی سازی و استفاده از این سیستم‌عامل در ایران مورد

نیاز است. برای اینکه این بحث عملیاتی تر باشد، اصطلاح مجری به شرکت (ها)، گروه (ها) و یا سازمانی اطلاق شده که در بومی سازی این سیستم عامل تلاش می کند و در اینجا سعی شده است، خط مشی مناسب برای بومی سازی این سیستم‌عامل ارائه شود.

۳-۲-۱- توزیع بومی

هر نرم‌افزار بومی، می‌باید در ایران دارای شبکه توزیع داخلی باشد و سیستم عامل بومی هم از این قاعده مستثنی نیست. سیستم‌عامل Enterprise بومی می‌باید به راحتی (رایگان یا در قبال اخذ هزینه) در دسترس کاربران عادی و مصرف کنندگان قرار گیرد. توزیع این سیستم‌عامل در شبکه توزیع فعلی برنامه‌ها در ایران از نکات بسیار حائز اهمیت می‌باشد.

مجری سیستم عامل بومی باید با توزیع مناسب این سیستم‌عامل، در سهولت دردسترس قرار گیری این سیستم‌عامل بکوشد. برای نمونه مجری با راه اندازی FTP Serverهای عمومی و مناسب به‌صورت استانی یا در دانشگاه‌ها در کنار سایر توزیع‌ها، می‌تواند امکان دریافت سیستم‌عامل را برای همگان فراهم آورد.

۳-۲-۲- پشتیبانی بومی

بطور کلی، کلیه نرم‌افزارهایی که پشتیبانی نمی‌شوند، قابل استفاده در سیستم‌های Enterprise نیستند. امروزه نرم‌افزارها و سیستم‌عامل‌های متنوع و Enterprise بسیاری در دنیا وجود دارند، مانندMicrosoft Windows، Microsoft SQL، Microsoft .Net ، Oracle و غیره که در راهکارهای Enterprise در سطح دنیا، قابل قبول هستند، ولی چون در ایران پشتیبانی نمی شوند، به هیچ عنوان قابل استفاده و اعتماد در سیستم‌های Enterprise بومی نیستند و اصولا یکی از شروط لازم Enterprise بودن یک نرم‌افزار، وجود امکانات قوی برای پشتیبانی مستمر آن است.

مجری سیستم‌عامل Enterprise، می‌باید کاملا مسلط به سیستم‌عامل لینوکس باشد و بتواند با قرار دادن راهکارهای مناسب برای پشتیبانی، شامل: تلفن، فکس، وب سایت، پست الکترونیکی و پست عادی، لیست‌های پست الکترونیکی و غیره، امکانات پشتیبانی در رده‌های مختلف را به مشتریان خود قرار دهد. با توجه به اینکه سیستم‌عامل Enterprise در کاربردهای Enterprise استفاده می‌شود، پشتیبانی از مشتریان و مصرف کنندگان این سیستم‌عامل به

دلایل مختلف، بسیار مهم است و مصرف کنندگان سیستم‌عامل Enterprise طبعا با مصرف کنندگان سایر سیستم‌عامل‌ها اختلاف دارند و مجری می‌باید با داشتن تیم پشتیبانی قوی و مجرب، به سرعت به نیازها و درخواست‌های مشتریان خود پاسخ دهد.

۳-۲-۳- به روزرسانی بومی

سیستم‌عامل مانند تمام برنامه‌ها و نرم‌افزارها توسعه و به‌روز می‌شود. همگام با به‌روز شدن تمام سیستم‌عامل‌ها و به‌وجود آمدن تکنولوژی‌های جدید و پدیدار شدن سخت‌افزارهای متنوع، سیستم‌عامل بومی هم می‌باید به‌روز شود و از تکنولوژی روز و سخت‌افزارهای جدید حمایت و پشتیبانی کند.

مجری سیستم‌عامل بومی، باید با توجه به حرکت سریع سیستم‌عامل لینوکس، بتواند خود را با این حرکت پیش ببرد و امکانات به‌روزرسانی را برای مشتریان و مصرف کنندگان سیستم‌عامل بومی فراهم سازد و سیستم به‌روز شدن سیستم‌عامل را به نحوی پیاده‌سازی کند تا با قطع و وصل شدن ارتباط شبکه ایران با اینترنت جهانی، سیستم به‌روز رسانی متوقف و دچار خلل نشود. از این رو مجری باید مجهز به تجهیزات سرور و عرض باند اینترنت مناسب با پشتیبان‌های کافی و مناسب به جهت سرویس دهی به مشتریان و مصرف کنندگان را در ایران دارا باشد و از برنامه‌های مناسب مانند YUM، UP2DATE و غیره که برای همین منظور طراحی شده‌اند، نهایت بهره را ببرد.

۳-۲-۴- تغییر، ارتقاء و تطبیق بومی

تمام حرکت نرم‌افزارهای دنیا بر پایه نیازهای به‌وجود آمده، سوار است. مجری سیستم‌عامل بومی باید دانش فنی لازم را داشته باشد تا با به‌وجود آمدن نیازهای خاص در میان مشتریان، سیستم‌عامل را با نیازهای مصرف کنندگان تطبیق کند. برای مثال ممکن است یکی از مصرف کنندگان سیستم‌عامل بومی، احتیاج به روال‌های امنیتی ویژه در لایه هسته سیستم‌عامل داشته باشد و مجری باید توان علمی و عملی پیاده سازی این روال‌ها را داشته باشد.

سیستم‌عامل بومی می‌باید نیازهای کشور عزیزمان ایران را در ارتباط با فناوری اطلاعات و امنیت ملی تامین سازد و در صورت پیش آمدن جنگ الکترونیکی، توان مقابله با نفوذ و خراب کاری اطلاعاتی را دارا باشد. این امر مهم فقط زمانی از مجری بر خواهد آمد که کاملا بر

سیستم‌عامل بومی، شبکه‌های کامپیوتری و مسائل امنیتی آنها تسلط داشته باشد و بتواند نسخه‌های ویژه با امکانات امنیتی شدید و قابل اعتماد تولید کند تا سازمان‌های دولتی و یا نظامی بتوانند به سهولت از این نسخه‌های خاص سیستم‌عامل بومی استفاده کنند.

۳-۲-۵- آموزش بومی

یکی از نکات بسیار مهم در استفاده از سیستم‌عامل بومی، آموزش این سیستم‌عامل در لایه‌های مختلف است. تا زمانی که آموزش سیستم‌عامل بومی به افراد ارائه نشده باشد نمی‌توان انتظار استفاده یا کاربرد این سیستم‌عامل را در سطوح مختلف داشت. سیستم‌عامل بومی می‌باید در میان کارشناسان و کاردانان این زمینه آموزش داده شود و ارائه این آموزش‌ها بهتر است با کمک وزارت آموزش و پرورش از سطوح متوسطه (دبیرستان) آغاز شود. اگرچه از مجری نمی‌توان انتظار داشت که در یک دوران کوتاه، در این زمینه انقلاب کند مگر اینکه مجری همان دولت باشد یا دولت او را به نحو مطلوب حمایت کند.

تشکیل کلاس‌ها و سمینارهای مختلف خصوصا، در سطح مدیران ارشد دولتی و خصوصی می‌تواند در امر آموزش بسیار مفید وقع شود. همچنین در حال حاضر هنوز دوره‌ها، سر فصل‌ها و کتب آموزشی هدف‌دار و مناسب به زبان فارسی برای آموزش لینوکس وجود ندارد که مجری می‌باید در این زمینه نیز نهایت تلاش خود را به کار گیرد.

۳-۲-۶- مستندات و راهنماهای بومی

سیستم‌عامل بومی می‌باید دارای مستندات فارسی و راهنماهای فارسی برای استفاده مصرف کنندگان و برنامه‌سازان داشته باشد. با توجه به اینکه برخی کارشناسان در نقاط مختلف کشور به زبان‌های خارجی چندان مسلط نیستند و هدف سیستم‌عامل بومی این است که همگان از این سیستم‌عامل بتوانند به سهولت استفاده کنند، می‌باید مستندات جامع و راهنماهای کامل از طریقه عملکرد این سیستم‌عامل وجود داشته باشد.

مجری سیستم‌عامل بومی وظیفه دارد تا موانع استفاده از سیستم‌عامل بومی را به حداقل برساند و یکی از موانع استفاده از یک سیستم‌عامل، عدم وجود مستندات و راهنماهای فارسی می‌باشد.

راهنماهای سیستم‌عامل بومی باید در قالب کتب چاپی، کتب الکترونیکی و مستندات قابل دریافت از اینترنت و وب سایت سیستم‌عامل بومی باشد. سیستم‌عامل لینوکس و برنامه‌های مهم و کاربردی آن که در سطح Enterprise هستند، هزاران خط مستندات به زبان انگلیسی دارند که مجری می‌باید به صلاحدید و اولویت بندی خود، به تدریج به تدوین و ترجمه این مستندات بپردازد تا کاربران و مصرف کنندگان به راحتی بتوانند از این مستندات نیز استفاده کنند.

۳-۲-۷- اطلاع رسانی، تبلیغ و فرهنگ سازی برای سیستم عامل بومی

چاپ مقالات آموزشی، وجود وب سایت، درج خبر در روزنامه‌ها و اخبار صدا و سیمای جمهوری اسلامی کمک به اطلاع رسانی و فرهنگ سازی برای استفاده از سیستم‌عامل بومی خواهد بود. مجری می‌تواند با ابزارهای ساده مانند ارائه خبر و مقالات مستمر در ارتباط با سیستم‌عامل بومی، کمک به فرهنگ سازی و استفاده از این سیستم‌عامل شود.

اطلاع رسانی و فرهنگ سازی، رمز موفقیت سیستم‌عامل بومی خواهد بود و تا زمانی که مردم و کارشناسان از وجود این سیستم‌عامل و ویژگی‌های آن اطلاع نداشته باشند، نمی‌توان انتظار داشت که از این سیستم‌عامل استفاده کنند.

۳-۲-۸- تقویم بومی

پیاده سازی تقویم بومی هم در لایه سیستمی میسر است و هم در لایه برنامه‌های کاربردی که مجری می‌باید به صلاح دید خود، امکانات استفاده از تقویم بومی را به طرق مختلف به کاربران ارائه کند. برای نمونه امکان ارائه تقویم فارسی هم در کتابخانه سیستمی Glibc ممکن است و هم به صورت یک کتابخانه خارجی و نوین که قابل استفاده در برنامه‌های زبان C و C++ باشد.

مجری در کنار اضافه کردن امکانات تقویم فارسی به این نکته نیز می‌باید توجه کند که سازگاری سیستم‌عامل را با سایر برنامه‌های موجود Enterprise در دنیا از بین نبرد و تقویم فارسی به عنوان یک گزینه قابل استفاده برای برنامه‌های بومی موجود باشد.

۳-۲-۹- فونت و نمایش صحیح فارسی

نمایش فونت و جملات فارسی در محیط گرافیکی سیستم‌عامل لینوکس بومی اولین قدم در کاربرد این سیستم‌عامل به عنوان یک سیستم‌عامل رومیزی است. اگرچه کاربرد سیستم‌عامل Enterprise بومی به عنوان محیط رومیزی به عنوان یک هدف اصلی مطرح نیست ولی قابلیت استفاده از این سیستم‌عامل به عنوان یک محیط رومیزی قدرتمند به مصرف کنندگان این سیستم‌عامل اجازه می‌دهد تا بتوانند کاربردهای معمول و ساده خود را مانند خواندن مستندات خود این سیستم‌عامل را با استفاده از امکانات خود این سیستم‌عامل انجام دهند.

۳-۲-۱۰- صفحه کلید فارسی

اکثر استفاده کنندگان سیستم‌عامل Enterprise، جامعه متخصصین هستند و مسائلی چون واژه پردازی و غیره در این سیستم‌عامل کمتر به چشم می‌خورد، در همین زمینه، اگرچه نیاز به پیاده سازی و پشتیبانی از صفحه کلید فارسی چندان در سیستم‌عامل Enterprise حاوی اهمیت جلوه نمی‌کند، ولی همواره می‌توان به آن به‌صورت یک گزینه برتری نگریست و در صورتیکه مجری بتواند با حفظ مسائل انطباق، امکانات صفحه کلید فارسی را به سیستم‌عامل به طور مناسب اضافه سازد، کمک به استفاده از این سیستم‌عامل به عنوان سیستم‌عامل رومیزی قدرتمند خواهد کرد.

۳-۲-۱۱- رابط کاربری فارسی

این امکان نیز در سیستم‌عامل Enterprise بومی حاوی اهمیت چندانی جلوه نمی‌کند و با مستند سازی فارسی و مناسب از یک رابط به زبان خارجی، می‌توان از این گزینه صرف نظر کرد. با این وجود، طراحی و تولید رابط فارسی در کنار رابط انگلیسی استاندارد می‌تواند به استقبال گسترده‌تر این سیستم‌عامل کمک کند و مجری می‌تواند همواره به عنوان یک گزینه مناسب، در صدد پیاده سازی رابط‌های مهم و پر کاربرد مانندGnome و KDE و برنامه‌های سیستمی برآید. برای نمونه، برنامه webmin اگرچه جزو سیستم‌عامل محسوب نمی‌شود، ولی یک برنامه بسیار قدرتمند برای تنظیم و کنترل سیستم‌عامل لینوکس است و می‌توان به سادگی امکانات فارسی را در آن به وجود آورد و به مصرف کنندگان این سیستم به عنوان یک

گزینه برتر و قابل استفاده معرفی کرد.(لازم به ذکر است، پروژه فارسی سازی این برنامه‌ها هم اکنون به کمک شورای عالی انفورماتیک و طرح ملی نرم‌افزارهای آزاد/ متن باز، در حال اجرا و پیشرفت است.)

۳-۳- بررسی سیستم عامل‌های لینوکس موجودEnterprise

برای شروع تولید و نگارش سیستم‌عامل Enterprise بومی می‌باید از یک سیستم‌عامل موجود Enterprise اقتباس و الگو برداری کرد. این اقتباس در درجه اول کمک می‌کند تا سیستم‌عامل بومی ایران هماهنگی و سازگاری را با یک سیستم‌عامل Enterprise جهانی داشته باشد و در شروع کار از استانداردهای Enterprise موجود فاصله نگیرد تا به تدریج مجری بتواند تسلط کافی را به اصول لازم در سیستم‌عامل‌های Enterprise پیدا کند.

در زیر سه سیستم‌عامل معروف و قابل استفاده برای سیستم‌عامل Enterprise بومی ذکر شده اند که هر سه برای الگو برداری مناسب هستند .

۳-۳-۱- سیستم عامل Red Hat Enterprise Linux

سیستم‌عامل RHEL معروفترین سیستم‌عامل Enterprise لینوکس در حال حاضر است و تقریبا تمام استانداردهای Enterprise لینوکس از ویژگی‌های این سیستم‌عامل منشق می‌شوند. سیستم‌عامل RHEL به خوبی توسط تمام برنامه‌های Enterprise موجود پشتیبانی می‌شود و از اعتبار جهانی مناسبی برخوردار است.

اشکال سیستم‌عامل RHEL در به‌روز نشدن این سیستم‌عامل است و برای استفاده از امکانات بروز رسانی موجود در این سیستم‌عامل، مصرف کننده می‌باید سالانه به شرکت Red Hat پول پرداخت کند. یکی از مزیت‌های شرکت Red Hat پایبند بودن این شرکت به قوانین متن باز ازجملهGPL است و سبب می‌شود تا متن (Source) تمام برنامه‌های موجود در این سیستم‌عامل را به‌طور رایگان در فضای اینترنت قرار دهد. با محیا بودن متن تمام برنامه‌ها، گروه‌هایی مانند CentOS وجود دارند که برنامه‌های سیستم‌عامل RHEL را کامپایل کرده و توزیع خود را با نام CentOS به طور رایگان در اختیار عموم قرار دهند. این توزیع به لحاظ ساختاری بسیار منطبق با سیستم‌عامل RHEL است و با مجوز GPL توزیع می‌شود.

سیستم‌عامل RHEL و توزیع‌هایی مانند CentOS شاید بهترین شروع برای سیستم‌عامل بومی باشند و با تغیر نام و نحوه بروز رسانی سیستم‌عامل، می‌توان این سیستم‌عامل را به سرعت به مرحله پایلوت سیستم‌عامل بومی رساند.

۳-۳-۲- سیستم عامل SUSE Linux Enterprise

سیستم‌عامل SUSE در اصل حاصل تلاش یک شرکت آلمانی بوده و در حال حاضر مالکیت آن در اختیار شرکت آمریکایی Novell است. این سیستم‌عامل جزو معروفترین سیستم‌عامل‌های Enterprise در زمینه لینوکس است و مانند همتای خود RHEL ازامکانات Enterprise بسیار مناسبی برخوردار است. محبوبیت و کاربرد این سیستم‌عامل در کشورهای اروپایی و سیستم‌های 64 بیتی (بخصوص AMD64) از RHEL بیشتر است. این سیستم‌عامل به همراه امکانات به‌روز رسانی آن، به‌طور سالیانه مانند RHEL به مشتریان فروخته می‌شود. سیستم‌عامل SUSE از امکانات بسیار غنی در پشتیبانی از سخت‌افزارها برخوردار است و محیط گرافیکی بسیار زیبا و کار آمد بخصوص در نسخه رومیزی را به کاربران خود ارائه می‌کند.

۳-۳-۳- سیستم عامل Ubuntu

سیستم‌عامل Ubuntu اگرچه آنچنان جزو سیستم‌عامل‌های Enterprise محسوب نمی‌شود و تبلیغات وسیعی مانند دو سیستم‌عامل ذکر شده فوق ندارد، ولی نگارش سرور آن، آنچنان کاربرد متفاوتی نسبت به سایر سیستم‌عامل‌های Enterprise ندارد و روز به روز به طرفداران آن افزوده می‌شود. این سیستم‌عامل که یک انشقاق از Debian است به خوبی توسط مجموعه Canonical پشتیبانی می‌شود و استفاده و به‌روز رسانی رایگانی دارد.

سیستم‌عامل Debian یکی از قدیمی ترین و پرطرفدارتین توزیع‌های لینوکس فعلی است که اگرچه پشتیبانی تجاری مستقیمی ندارد، ولی بسیاری از شرکت‌های معتبر از آن حمایت می‌کنند و منابع و مقالات متنوع و گوناگونی برای این سیستم‌عامل وجود دارد که از محاسن این سیستم‌عامل است.

۳-۴- نتایج

دراین قسمت ویژگی‌های مطلوب و مورد نیاز سیستم‌عامل Enterprise بومی را به تفکیک آورده و از دو سیستم‌عامل Enterprise در سطح جهان نام برده است که برای تولید سیستم‌عامل Enterprise بومی قابل استفاده و الگو برداری هستند.

دقت شود که بومی سازی سیستم‌عامل Enterprise با بومی سازی سیستم‌عامل رومیزی که شامل اضافه کردن فونت و رابط کاربری فارسی و غیره است، بسیار متفاوت است و این سیستم‌عامل غالبا در محیط‌های شبکه و اماکنی که معمولاً در دیدکاربران عادی نیستند، استفاده می‌شود و این سیستم‌عامل‌ها معمولا همه روزه در حال سرویس دادن و زیر بار سنگین شبکه و غیره استفاده می‌شوند.

در سیستم‌عامل Enterprise بومی، رابط کاربری فارسی و امکانات نظیر آن اگرچه اهمیت دارند، ولی نسبت به ویژگی‌های مورد انتظار برای یک سیستم‌عامل Enterprise مانند پشتیبانی مستمر و بروز رسانی ، از اهمیت کمتری برخوردارند و قابل صرف نظر هستند.

فصل چهارم – مشخصات سیستم عامل های بلادرنگ و تعبیه شده

کامپیوتر در همه زمینه‌های زندگی امروزه نفوذ کرده است و کمتر کسی است که از این ابزار اطلاعی نداشته باشد. هرچند این ابزار در اصل فقط یک ماشین منطق و محاسب قدرتمند می باشد ولی به لحاظ ساختاری و منطق برنامه پذیر آن، قدرت انطباق و حل بسیاری از مشکلات امروزی ما را دارد.

کامپیوترهای مختلف در تلویزیون‌ها، هواپیماها، موشک‌های نظامی، خودروها و بسیاری مصارف دیگر وجود دارند و بی شک همه این کامپیوترها در هر نقطه ای که هستند، طراحی‌های ویژه مناسب محل استقرار خود دارند و مجهز به برنامه‌های مخصوص هستند. طبعا سیستم عامل‌های این کامپیوترها نیز با یکدیگر مختلف است و عملکردهای متنوعی را ارائه می‌کنند.

در این قسمت سیستم عامل‌های بلادرنگ و سیستم عامل‌های تعبیه شده که دو فاز بسیار مهم و نزدیک به هم در مقوله سیستم عامل‌ها هستند، بررسی و تعریف شده است.

۴–۱– مفاهیم سیستم عامل بلادرنگ

در بعضی زمینه ها، و مسائل روزمره، زمان نقش مهمی را ایفا می کند و از اهمیت ویژه‌ای برخوردار است. برای یک مثال بسیار ساده، برای پخت یک غذا، جدا از شیوه پخت، برنامه زمانبندی خاصی وجود دارد که باید آشپز آن را رعایت کند و در صورتیکه این برنامه زمانبندی رعایت نشود و توالی پخت مواد غذایی و زمان بندی های لازم رعایت نشوند، مسلما غذای پخته شده از کیفیت و مزه لازم برخوردار نخواهد بود.

به عنوان یک مثال صنعتی و علمی، یک ماشین تزریق پلاستیک را تصور کنید، این ماشین در یک بازه زمانی خاص قالب تزریق پلاستیک را می بندد، در بازه بعدی پلاستیک مذاب را به داخل قالب تزریق می کند و پس از اتمام تزریق، قالب را باز کرده و صبر می کند تا اپراتور محصول تولید شده پلاستیکی را خارج کند. در این مثال، دستگاه تزریق، زمانبندی بسیار دقیقی را رعایت می کند و در این حین از کمک سنسورها هم نهایت استفاده را می برد تا این زمانبندی با حرکت فیزیکی ماشین منطبق باشد. اگر زمانبندی این دستگاه به نحوی مختل شود که قبل از بسته شدن قالب، پلاستیک را تزریق کند یا قبل از اتمام تزریق قالب را باز کند،

هیچ محصولی تولید نخواهد شد و پلاستیک مذاب به دستگاه و احتمالا اپراتور صدمه خواهد زد.

در دنیای کامپیوتر هم بعضی از برنامه‌ها می باید در زمانبندی خاص خود اجرا شوند و خروجی‌های لازم را تولید کنند. برای مثال کامپیوتری که آشپزی می کند یا کامپیوتری که دستگاه تزریق پلاستیک یا دستگاه تراش و متشابهات آنها را کنترل می کند، می باید در زمانبندی خاص خود، عملکرد مخصوصی را انجام دهد.

با توجه به اینکه در یک کامپیوتر، اختیار کلی در دست سیستم عامل است، یک برنامه به تنهایی نمی تواند کنترل دقیقی بر اجرا و زمانبندی خود داشته باشند، برای حل مشکل زمانبندی، سیستم عامل‌هایی به وجود آمدند که پارامتر زمان در آنها یک فاکتور اساسی و با اهمیت است و به کمک این سیستم عامل‌ها که به سیستم عامل‌های بلادرنگ معروفند، برنامه‌ها می توانند در زمان‌های لازم وظایف خود را انجام دهند.

سیستم عامل‌های بلادرنگ مشابه سیستم عامل‌های معمولی هستند با این تفاوت که زمان عملکرد و پاسخگویی سیستم عامل و برنامه در آن بسیار با اهمیت است و بطور خلاصه زمان پاسخ گویی به عنوان مهمترین فاکتور محسوب می شود. در سیستم عامل‌های امروزی که معمولا دارای تکنیک اشتراک زمانی هستند، یک برنامه خود به خود نمی تواند بطور دقیق خود را با ساعت سیستم هماهنگ کند و نمی تواند سر ساعت خاصی یک سیگنال ویژه به درگاه‌های خروجی ارسال کند یا پردازش خاصی را انجام دهد، علت این امر وجود برنامه بزرگتری به نام سیستم عامل است که کنترل اصلی سخت‌افزار کامپیوتر را در دست می گیرد. سیستم عامل تمام درگاه‌های ورودی و خروجی و وقفه‌های سیستم از جمله وقفه ساعت را در دست دارد و برنامه‌ها نمی توانند نظارت درستی بر روی وقفه ساعت کامپیوتر و سایر وقفه‌ها داشته باشند. از طرف دیگر سیستم عامل‌های امروز که از تکنیک اشتراک زمانی و حافظه مجازی استفاده می‌کنند می توانند بنا به صلاح دید خود برنامه و اطلاعات آن را از روی حافظه اصلی به حافظه مجازی که معمولا دیسک سخت است، منتقل کنند و برنامه دیگری را بنا به صلاح دید خود اجرا کنند، این سبب می شود تا درست در لحظه‌ای که لازم است یک برنامه پاسخ دهد، زمانی برای انتقال مجدد برنامه و اطلاعات آن از حافظه مجازی به حافظه اصلی تلف شود و سپس کنترل کامپیوتر به برنامه اصلی مورد نظر منتقل شود و در نهایت برنامه در زمان مناسب پاسخ لازم را ارائه ندهد.

پاسخ تمام مشکلات فوق سیستم عامل‌های بلادرنگ هستند. این سیستم‌ها با طراحی خاص خود قدرت لازم را به برنامه‌های خود می دهند تا در زمان مناسب بتواند پاسخ مناسب را صادر کند.

تعریف عمومی سیستم عامل بلادرنگ عبارتست از سیستم عاملی که می تواند در بازه زمانی مشخص، عملکرد مشخص و تعریف شده‌ای را انجام دهد. این تعریف بیانگر این است که یک سیستم عامل بلادرنگ می تواند یک کار را در ۵ ساعت یا در ۱ میلی ثانیه بسته به طراحی انجام دهد و از آنجا که هر دو سیستم عامل در زمان مشخص خود پاسخ می دهند، هر دو بلادرنگ محسوب می‌شوند و تعریف بلادرنگی با مفهوم سریع بودن سیستم‌عامل یکسان نیست.

سیستم‌های بلادرنگ معمولا در سیستم‌های واکنش دار یا سیستم‌های تعبیه شده، استفاده می شوند. سیستم‌های واکنش دار سیستم‌هایی هستند که به ورودی‌های خود مانند کلیدها و سنسورها واکنش‌هایی مناسب با زمان نشان می دهند، برای مثال سیستم‌های کنترل اطفاء حریق به ورودی های خود مانند سنسورهای حرارتی یا دگمه‌های خطر پاسخ می دهند و سیستم در حداقل زمان برنامه ریزی شده به کار می افتد.

سیستم‌های بلادرنگ به سه دسته سیستم‌های بلادرنگ سخت[1]، سیستم‌های بلادرنگ استوار (یا محکم)[2] و سیستم‌های بلادرنگ نرم[3] تقسیم می شوند، در بعضی تعاریف معمول دیگر سیستم‌های بلادرنگ استوار، جزو سیستم‌های بلادرنگ نرم محسوب می شوند و سیستم‌های بلادرنگ فقط به دو دسته تقسیم می شوند. درادامه سیستم‌های بلادرنگ بطور مفصل در سه دسته بررسی می کنیم.

۴-۱-۱- سیستم عامل های بلادرنگ نرم

سیستم‌های بلادرنگ نرم سیستم‌هایی هستند که خطاهای متوالی کوچک زمانی در این سیستم‌ها قابل پذیرش و بخشش است. این سیستم‌ها معمولا در نقاط غیر بحرانی استفاده می‌شوند. برای نمونه چندین تاخیر یا قطع یک میلی ثانیه ای در پخش صدا یا تصویرهای متوالی در سیستم های پخش کننده مانند رادیو و تلویزیون ضرر آنچنانی وارد نمی سازد.

---

1 Hard Real Time Systems
2 Firm Real Time Systems
3 Soft Real Time Systems

وجود راندمان خطای قابل بخشش سبب می شود تا سیستم‌های بلادرنگ نرم از لحاظ پیاده سازی بسیار ساده تر از سیستم عامل‌های بلادرنگ سخت و سیستم‌های بلادرنگ استوار باشند. این سیستم عامل‌ها معمولا بسیار نزدیک به سیستم‌های عامل های معمولی هستند با این تفاوت که یک برنامه خاص با اولویت بالا توسط سیستم عامل در حال اجراست، بطوریکه این برنامه هیچ وقت از حافظه خارج نمی شود و اولویت اجرای آن برابر یا نزدیک به اولویت اجرای خود سیستم عامل است.

۴-۱-۲- سیستم عامل های بلادرنگ استوار

سیستم‌های بلادرنگ استوار سیستم‌هایی هستند که خطاهای کوچک و نادر دراین سیستم‌ها قابل پذیرش و بخشش است ولی تکرار این خطاها و افزایش آنها غیر قابل پذیرش است. این سیستم‌ها معمولا در نقاط نسبتا بحرانی استفاده می شوند. برای نمونه تاخیر کوچک یک روبات در خط تولید ممکن است ضربه چندانی به محصول تولید شده و عملکرد سایر روباتها وارد نسازد ولی اگر این تاخیر و خطاها مرتب تکرار و زیاد شود، محصول تولید شده ممکن است از کیفیت لازم برخوردار نباشد.

وجود این راندمان خطای کوچک قابل بخشش سبب می شود تا سیستم‌های بلادرنگ استوار از لحاظ پیاده سازی ساده تر از سیستم عامل‌های بلادرنگ سخت باشند. این سیستم عامل‌ها همچنان نزدیک به سیستم عامل‌های معمولی هستند و یک برنامه خاص در آنها با اولویت بسیار بالا توسط سیستم عامل در حال اجراست و این برنامه هیچ وقت از حافظه خارج نمی شود و گاهی اولویت اجرای آن از خود سیستم عامل بالاتر است.

۴-۱-۳- سیستم عامل های بلادرنگ سخت

سیستم‌های بلادرنگ سخت سیستم‌هایی هستند که خطاهای کوچک زمانی در این سیستم‌ها اصولا قابل پذیرش و بخشش نیست. این سیستم‌ها معمولا در نقاط بسیار بحرانی استفاده می شوند. برای نمونه تاخیر یک میلی ثانیه ای یک کامپیوتر کنترل کننده هواپیما، ممکن است منجر به مرگ تمام سرنشینان آن شود. معمولا در تمام سیستم‌های نظامی و سیستم‌هایی که ممکن است سلامتی انسان‌ها را به خطر بیندازند، مانند سیستم‌های ناوبری هواپیماها، تجهیزات پزشکی و غیره،از سیستم‌های بلادرنگ سخت استفاده می کنند.

سیستم عامل‌های بلادرنگ سخت در پیاده سازی گاهی بسیار متفاوت تر از سیستم عامل‌های معمولی و سیستم عامل های بلادرنگ استوار و سیستم عامل های بلادرنگ نرم هستند، بطوریکه گاهی مرز مشخصی بین سیستم عامل و برنامه در آن وجود ندارد و همه سیستم یکپارچه ساخته شده است. در بعضی موارد بیشتر کد سیستم عامل‌های سخت برای رسیدن به حداکثر بازدهی به زبان اسمبلی یا زبان‌های بسیار سطح پایین دیگر نگارش می‌شود.

در بعضی سیستم عامل‌های بلادرنگ سخت معمولا تکنیک‌های اشتراک زمانی، حافظه مجازی و حتی مدیریت فایل‌ها نیز وجود ندارد، از این رو گاهی این سیستم عامل‌ها را زیر مجموعه سیستم عامل‌ها قید نمی کنند و بیشتر حالات خاص برنامه نویسی محسوب می‌شوند و در کتب آکادمیک دانشگاهی، زیاد درباره آنها بحث نمی شود.

۴-۲- مفاهیم سیستم عامل تعبیه شده

کامپیوترهای کوچک و معمولا مخفی از دید کاربران، در همه زمینه‌ها منجمله اتومبیل، تلوزیون و در بسیاری نقاط دیگر استفاده می شوند. طراحی این کامپیوترهای کوچک گاهی به جز اختلاف در دستگاه‌های ورودی و خروجی و شکل و شمایل ویژه خود، هیچ تفاوتی با کامپیوترهای معمولی ندارند. این کامپیوترهای کوچک و مخفی، سیستم عامل‌های مخصوص و برنامه‌های ویژه خود را دارند.

سیستم عامل‌های تعبیه شده ممکن است خاصیت‌های سیستم عامل‌های بلادرنگ را نیز دارا باشند ولی در تعریف با سیستم عامل‌های بلادرنگ اختلاف دارند.سیستم عامل تعبیه شده در تعریف،سیستم عاملی است که در یک کامپیوتر تعبیه شده که در یک سیستم بزرگتر عمل می کند. این سیستم عامل بزرگتر ممکن است یک سیستم مکانیکی مانند خودرو یا سیستم‌های الکتریکی مانند تلوزیون و غیره باشد. در زیر به ویژگی‌های این سیستم‌ها اشاره شده است.

۴-۲-۱- واسط کاربری بسیار ساده

سیستم عامل‌های این کامپیوترهای کوچک، بر عکس سیستم عامل‌های معمولی کامپیوترهای شخصی که کاربرد عمومی دارند و با کاربران مرتب در ارتباط هستند، فقط توان اجرای کاربردهای محدود و از پیش تعریف شده‌ای را دارند که معمولا به مدیریت انسانی یا

حضور یک کاربر احتیاجی نیست. دراین سیستم‌ها محاوره با کاربر یا وجود ندارد؛ یا بسیار محدود است و مسائلی چون واسط‌های کاربری زیبا در آنها کمتر به چشم می خورد.

۴-۲-۲- بهینه سازی گسترده

برنامه‌های سیستم‌های تعبیه شده و خود سیستم عامل در این دستگاه‌ها، با نهایت دقت نگارش و بهینه می‌شوند، از این رو در این سیستم‌ها معمولا بار پردازشی بسیار کمتری نسبت به سیستم‌های با کاربردهای عمومی وجود دارد واین سیستم‌ها به پردازنده‌ای به مراتب ضعیف تر از پردازنده‌های دستگاه‌های عمومی احتیاج دارند.

۴-۲-۳- اشکالات نرم‌افزاری کمتر

به دلایل فقدان واسط کاربری بزرگ، پشتیبانی کمتر از سخت‌افزارها و امکانات محدود ارائه شده در سیستم‌های تعبیه شده، سیستم عامل های تعبیه شده بسیار بهینه تر و سبک تر از سیستم عامل های عمومی هستند. سبک بودن این سیستم عامل و کوچک بودن کد منبع آن، سبب می‌شود تا این سیستم عامل‌ها کمتر از سیستم عامل‌های عمومی که میلیونها خط کد منبع دارند، دچار مشکلات نرم‌افزاری شوند.

۴-۲-۴- سخت‌افزار ساده و ارزان

این سیستم‌ها به دلایل فوق معمولا احتیاجی به پردازنده بسیار قوی و حافظه فراوان ندارند، در آنها دیسک‌های سخت و گرداننده‌های دیسک‌های نرم و گرداننده لوح‌های فشرده یا اتصالات شبکه کمتر به چشم می خورد. ساده بودن سخت‌افزار و سبک بودن آن سبب می‌شود تا قیمت سخت‌افزار این سیستم‌ها بسیار ارزان‌تر از سیستم‌های معمولی شود و بتوان از آنها بطور گسترده در صنعت و مصارف دیگر استفاده کرد.

۴-۲-۵- مدیریت بهتر سخت‌افزار

همانطور که ذکر شد، سخت‌افزار سیستم عامل‌های تعبیه شده ساده تر و محدودتر از سخت‌افزار کامپیوترهای معمولی است و کارایی کمتری دارد، از این رو سیستم عامل‌های تعبیه شده مدیریت بهتری بر سخت‌افزار خود دارند. برای نمونه کامپیوتر یک خودرو که وظیفه تنظیم موتور را دارد، مانیتور، کارت گرافیکی، کارت صدا، کارت شبکه، صفحه کلید و امثال اینها را

ندارد و سیستم عامل احتیاجی به مدیریت این سخت‌افزارهای اضافی را ندارد و سخت‌افزارهای محدود خود شامل سنسورها و سایر ورودی/خروجی‌های محدود خود را بهتر مدیریت می کند.

۴-۳- مفاهیم سیستم عامل های بلادرنگ و تعبیه شده

اگر یک کامپیوتر و برنامه های آن هم خاصیت بلادرنگی و هم به صورت تعبیه شده در یک سیستم بزرگتر باشد، سیستم عامل آن سیستم عامل بلادرنگ تعبیه شده خواهد بود. مثال این سیستم‌ها فراوان هستند و اصولا بسیاری از کامپیوترهای صنعتی، دارای سیستم عامل بلادرنگ تعبیه شده هستند.

سیستم‌های ناوبری هواپیما، سیستم کنترل دستگاه تزریق، سیستم کنترل هدایت موشکی و بسیاری از مثال‌های قید شده دیگر در این گزارش و سیستم‌های صنعتی دیگر، هیچکدام شبیه به کامپیوترهای عمومی نیستند و به صورت تعبیه شده در یک مجموعه بزرگتر استفاده می شوند از این رو همگی سیستم‌های تعبیه شده هستند. از طرف دیگر چون در این سیستم‌ها شاخص‌ها و پارامترهای مخصوص زمانی هم به چشم می خورند، همگی بلادرنگ نیز محسوب می شوند.

اصولا مفهوم تعبیه شده بیشتر اشاره به سخت‌افزار و مفهوم بلادرنگی بیشتر اشاره به نقش و عملکرد سیستم و برنامه های آن دارد.

۴-۴- هسته سیستم عامل های بلادرنگ - تعبیه شده

پروسس[1] جزئی از یک برنامه در حال اجراء است که سیستم عامل طبق یک برنامه زمانی آن را اجراء می کند. معمولا پروسس‌ها به صورت یک ساختمان داده ها[2] و حداقل یک وضعیت اجرایی، شناخته می شوند و هویت، صفات و منابع تخصیص داده شده خود را دارند. ریسمان[3] حالت سبک یک پروسس است که با ریسمان های دیگر یا پروسس‌های دیگر منابع خود را به

1 Process
2 Data Structure
3 Thread

اشتراک قرار می دهد. هر ریسمان در یک پروسس قرار دارد و ریسمان‌های یک پروسس منابع مشترکی دارند.

تمام سیستم عامل‌ها و سیستم عامل‌های بلادرنگ باید در ابتدا ، سه عملکرد و وظیفه ۱- زمانبندی[1] ،۲- اعزام[2] ۳-ارتباط داخلی[3] و همگامی[4] را فراهم آورند. هسته سیستم عامل کوچکترین واحدی است که می باید این سه وظیفه را انجام دهد.

واحد زمانبندی هسته تصمیم می گیرد که کدام فعالیت و برنامه در یک سیستم چندکاره[5] انجام شود، در حالیکه واحد اعزام کننده عملیات ساماندهی لازم را برای اجرای آن برنامه فراهم می سازد و واحد ارتباط داخلی و همگامی، اشتراک مساعی کردن برنامه‌ها را با یکدیگر فراهم می آورد. جدول (۴-۱) ارتباط داخلی واحدهای مختلف یک هسته و انواع هسته های را بسته به عملکرد نمایش می دهد. با حرکت از سمت پایین به بالا، هسته سیستم عامل امکانات کاملتر و متنوع‌تری را ارائه می کند.

هسته های Nanokernel فقط امکانات مدیریت ریسمانها -که در بالا گفته شد را ارائه می‌دهند در حالیکه هسته های Microkernel علاوه بر امکانات هسته های Nanokernel، امکانات مدیریت وظایف را نیز دارند.هسته‌های معمول(Kernel) ارتباطات و همگامی پروسس‌ها را نیز مدیریت می کنند. هسته‌های Executive از مدیریت حافظه و مدیریت دستگاه‌های ورودی و خروجی نیز برخوردارند. با توجه به تقسیم بندی بالا، تقریبا تمام سیستم عامل‌های همه کاربره و عمومی و اکثر سیستم عامل‌های بلادرنگ تجاری از نوع Executive هستند.

جدول۴-۱: ارتباط داخلی واحد های مختلف یک هسته

Operating System	پوسته‌ها و رابط های کاربری
Executive	پشتیبانی از فایلها ، دیسک‌ها و ورودی/خروجی ها
Kernel	ارتباطات و همگامی پروسس ها
Microkernel	مدیریت وظایف [6]
Nanokernel	مدیریت ریسمانها

1 Scheduling
2 Dispatching
3 Intercommunication
4 Synchronization
5 Multitasking
6 Task Scheduling

۴-۴-۱- هسته های کاذب[1]

در بعضی سیستم‌های بلادرنگ یا سیستم‌های تعبیه شده، می توان از مسائلی چون وقفه‌ها و حتی از خود سیستم عامل صرفنظر کرد. این سیستم‌ها برای پیاده سازی و آنالیز بسیار ساده تر هستند و نمونه از پیاده سازی‌های این سیستم‌ها شامل وجود یک حلقه بی‌انتها در بدنه اصلی برنامه و انجام عملیات مختلف برنامه در خلال این حلقه است. این برنامه هم سیستم عامل و هم برنامه اجرایی محسوب می شود. کد مثال زیر یک نمونه از پیاده سازی این گونه سیستم عامل‌ها است:

```
for(;;) { /* do forever */
if (packet_here) /* check flag */
{
process_data(); /* process data */
packet_here=0; /* reset flag */
}
```

۴-۴-۲- سیستم عامل های مبتنی بر وقفه ها[2]

سیستم عامل‌های مبتنی بر وقفه‌ها بر اساس رخ دادن انواع وقفه‌های نرم‌افزاری و سخت‌افزاری کار می کنند. در این سیستم عامل‌ها، بدنه سیستم عامل معمولا خود به خود کاری را انجام نمی دهد و با رخ دادن یک وقفه سخت‌افزاری و یا نرم‌افزاری، سیستم عامل روتین مناسب وقفه را که عملیات و وظیفه اصلی سیستم عامل در آن نگارش شده است، را اجرا می کند.

کلا سیستم عامل‌های مبتنی بر وقفه‌ها در سیستم عامل‌های تعبیه شده و سیستم هایی که نقش کنترل‌گرها را دارند بسیار استفاده می شوند. در سخت‌افزارهای سیستم‌های تعبیه شده تعداد وقفه‌ها، ماهیت و اولویت آنها طبق طراحی سخت‌افزاری و عملکرد سیستم، مشخص است. برای مثال در تعیین اولویت وقفه‌ها، معمولا طراحان سخت‌افزار سیستم وقفه شماره یک را با اولویت بیشتری نسبت به وقفه شماره دو و سایر وقفه‌ها قرار می دهند و سیستم نرم‌افزاری با کمک سخت‌افزار ویژه با پیاده سازی مخصوصی که دارد، در نظارت بر عملکرد وقفه‌ها با توجه به اولویت آنها تصمیم می گیرد. در این سیستم‌ها اگر وقفه‌ها بطور همزمان رخ دهند، سیستم

---

1 Pseudo Kernels
2 Interrupt-Driven

عامل در ابتدا عملیاتی را که در طراحی با اولویت بیشتری منظور شده، اجرا می‌کند و درصورتیکه وقفه‌ها با اولویت یکسان باشند، سیستم‌عامل می‌تواند با الگوریتم گردش نوبتی[1] یا الگوریتم اولین ورود، اولین پاسخ[2] به وقفه های همزمان جواب مناسب بدهد.

کد مثال زیر یک نمونه بسیار ساده از پیاده سازی سیستم عامل‌ها مبتنی بر وقفه است. دقت کنید که در کد زیر فضای کاری پردازنده (Context) مانند رجیسترها، در ابتدای وقفه ذخیره شده و در انتها به حالت اولیه باز می گردد. این سبب می شود تا در انتهای روتین وقفه، پردازنده بتواند مجددا به کار قبلی خود قبل از رخ دادن وقفه بپردازد. عملکرد این سیستم عامل مشابه الگوریتم آخرین ورود، اولین خروج[3] است.

```
void main(void)
/*initialize system, load interrupt handlers */
{
init();
while(TRUE); /* infinite wait loop */
}
void intl (void) /* interrupt handler 1 */
{
save(context); /* save context on stack */
task1(); /* execute task 1 */
restore(context); /* restore context from stack */
}
void int2(void) /* interrupt handler 2 */
{
save(context); /* save context on stack */
task2(); /* execute task 2 */
restore(context); /* restore context from stack */
}
```

۴-۴-۳- سیستم عامل های دارای اولویت انحصاری[4]

در سیستم عامل هایی که یک یا چند عملکرد مانند رخ دادن یک وقفه یا عملیات مشابه دارای اولویت خاصی نسبت به همه عملکردها است و یا عملکردهای سیستم عامل به صورت ترتیبی نسبت به یکدیگر اولویت دارند، سیستم عامل های دارای اولویت انحصاری نامیده

1 Round Robin
2 First Come First Served
3 Last In First Out ,LIFO
4 Preemptive-Priority

می‌شوند. این سیستم عامل‌ها پیاده سازی متفاوتی با سیستم عامل های دیگر دارند و در این سیستم عامل‌ها با پیش آمدن زمان اجرای عملکرد اولویت دار مانند رخ دادن یک وقفه بسیار مهم در سیستم، تمام عملکردهای سیستم و حتی سایر وقفه‌ها تا اتمام عملیات انحصاری، متوقف می شوند.

نمونه این سیستم‌ها ، برنامه کنترل کننده یک نیروگاه اتمی است. در نیروگاه‌های اتمی، کنترل دمای نیروگاه از هر عملکردی، برای کامپیوتر کنترلگر تعبیه شده مهمتر است و سیستم تعبیه شده و برنامه آن در صورت دریافت وقفه مربوط به دمای نیروگاه، تمام کارهای خود را متوقف می کنند و اجازه نمی دهند تا هیچ وقفه دیگری اتفاق بیفتد و سریعا به کنترل درجه دما و عملیات مربوطه می‌پردازد.

۴-۴-۴- سیستم های ترکیبی [1]

سیستم عامل‌هایی که به صورت ترکیب مشخصات در سیستم عامل‌های مبتنی بر وقفه و سیستم عامل‌های دارای اولویت انحصاری هستند، سیستم عامل‌های ترکیبی نامیده می‌شوند. در این سیستم عامل‌ها بعضی از روتین‌ها بسیار با اهمیت هستند و لذا اولویت بیشتری دارند، در صورتیکه بعضی روتین‌های دیگر دارای اهمیت یکسانی هستند از نظر اولویت در یک رده قرار می گیرند. در سیستم عامل‌های ترکیبی، در صورت رخ دادن همزمان یک یا چند وقفه، سیستم عامل ابتدا با توجه به اولویت آنها، روتین مربوطه را اجرا می کند و در صورتیکه وقفه‌های پیش آمده، اولویت یکسانی داشتند، معمولا با کمک الگوریتم گردش نوبتی یا الگوریتم اولین ورود، اولین پاسخ ، روتین لازم را اجرا می کند.

۴-۴-۵- سیستم های دارای بلوک کنترل وظیفه [2]

مدرن ترین سیستم عامل‌های بلادرنگ و تعبیه شده، سیستم عامل های دارای بلوک کنترل وظیفه (برنامه) هستند. در این سیستم عامل‌ها، مشخصات تمام وظایف سیستم عامل و برنامه‌ها در لیست‌های پیوندی نگهداری می شوند. هر وظیفه دارای یک مشخصه عددی، یک اولویت

---

1 Hybrid
2 Task-Control Block

اجرایی، یک وضعیت اجرایی و مقادیر دیگری است که مدیر کنترل کننده وظایف با توجه به اولویت اجرای این برنامه‌ها و وضعیت اجرایی آنها، اجرای برنامه‌ها را کنترل می کند.

۴-۵- ساختار داخلی سیستم عامل و استاندارد POSIX [1]

برای شناخت سیستم عامل و مفاهیم آن، می باید نحوه ارتباط سیستم عامل با برنامه‌های کاربردی را نیز بررسی کرد. ارتباط سیستم عامل با برنامه‌های کاربردی به صورت یک سری دستور العمل واسط است که سابقاً این ارتباط به نام فراخوانی سیستمی (System Calls) معروف بود. برای درک صحیح سیستم عامل، شناخت این ارتباط و نحوه آن امری ضروری است. اگرچه این ارتباط در سیستم عامل‌های مختلف، یکسان نیست ولی مفاهیم کلی آن در همه سیستم عامل‌ها مشترک است. در زمینه ارتباطات سیستم عامل با برنامه‌های کاربردی، استانداردهایی منجمله استاندارد POSIX وجود دارد که سعی در استاندارد کردن و یکسان کردن ارتباط سیستم عامل با برنامه‌های کاربردی دارند. بسیاری از سیستم عامل‌های معروف مانند یونیکس، لینوکس، خانواده BSD، ویندوز (تا حدودی) و سیستم عامل‌های تجاری برای سیستم‌های بلادرنگ یا تعبیه شده مانند QNX به خوبی از این استاندارد پشتیبانی می کنند.

یکسان شدن این ارتباط و تبعیت سیستم عامل‌ها از این استانداردها به برنامه نویس این امکان را می دهد که برنامه خود را در یک سیستم عامل واجد شرایط این استاندارد نوشته و برنامه خود را به سادگی و بدون دغدغه به بقیه سیستم عامل‌های تابع این استاندارد انتقال دهد.

در استاندارد POSIX، واسط یک سیستم عامل استاندارد و محیط آن[2] (که همگی به زبان C هستند)، به همراه یک پوسته[3] و تعدادی برنامه کاربردی، که کمک می کند تا برنامه‌ها در لایه کد منبع[4] قابل انتقال باشند، تعریف و استاندارد شده است. این استانداردها از یک پروژه در حدود سال ۱۹۸۵ پدیدار شدند و همچنان در حال تکامل هستند. در حال‌حاضر (بهار ۱۳۸۵)، نسخه سال ۲۰۰۴ این استانداردها مطرح است که IEEE و Open Group به صورت مشترک

1 Standard for Information Technology - Portable Operating System Interface (POSIX)
2 Standard Operating System Interface and Environment,
3 Shell
4 Source Code

در کمیته ای به نام PASC [1] به همراه سایر اعضا و علاقمندان در این زمینه، این استانداردها را تنظیم می کنند. نام POSIX را ریچارد استالمن در درخواست IEEE برای نامی که در اذهان می ماند، پیشنهاد کرده است.

در حال حاضر این استاندارد در حدوداً ۱۵ سند مختلف تعریف شده است و به لحاظ موضوعی به سه بخش عمده زیر تقسیم می‌شود:

- توابع مربوط به هسته سیستم عامل که در آن استاندارد POSIX.1 ، سرویس های بلادرنگ و توسعه های آن، واسط چند ریسمانی، واسط امنیتی و شبکه‌ها ، هم وجود دارند.
- دستورات و برنامه‌های کاربردی
- آزمون‌های بررسی تطبیق با این استاندارد

سیستم عامل‌های لینوکس، خانواده BSD و حتی سیستم‌های آکادمیک مانند MINIX (تا حدودی) و MINIX 3 از این استاندارد پشتیبانی می کنند و این سبب می شود تا بتوان کد برنامه‌های زبان C را به این سیستم‌ها انتقال داد. با وجود این استانداردها، نگارش برنامه‌های مختلف بلادرنگ، بسیار ساده تر می شود. بخصوص که POSIX نکات و استانداردهایی را در ارتباط با استانداردهای برنامه‌های بلادرنگ و ویژگی‌های سیستم عامل‌های بلادرنگ را در خود جا داده است.

۴-۶- بررسی مؤلفه های سیستم عامل های بلادرنگ و تعبیه شده

سیستم عامل‌ها را می توان به اجزای کوچکتری تقسیم کرد و هر بخش را بطور مستقل بررسی کرد. از آنجا که بررسی تفصیلی این بخش‌ها و الگوریتم‌های پیاده سازی آنها از حیطه‌ی این بحث خارج است فقط به آنها اشاره شده است و بیشتر به اختلافات مفاهیم کلاسیک بین سیستم عامل‌های همه کاربره و سیستم‌های عامل بلادرنگ و تعبیه شده ، اشاره شده است.

---

1 Www.pasc.org

۴-۶-۱- مدیریت پردازش

هر برنامه از یک یا چند پروسس تشکیل شده است. پروسس‌ها نیاز به منابع سیستمی مانند پردازنده، حافظه و ... دارند و با یکدیگر ممکن است ارتباط داشته باشند. مدیریت این پروسس‌ها و تخصیص منابع به آنها توسط سیستم عامل صورت می‌گیرد. در سیستم‌عامل‌های بلادرنگ، مدیریت پروسس‌ها باید توانایی ارائه خدمات به پروسس‌ها را در بازه زمانی مناسب، طبق درخواست پروسس ارائه دهد. این مدیریت می تواند از الگوریتم گردش نوبتی یا الگوریتم اولین ورود، اولین پاسخ، و الگوریتم های پویای اولویت پروسس بر اساس نزدیک ترین زمان پاسخ درخواست شده[1]و غیره بر اساس نوع سیستم و عملکرد آن، استفاده کند.

۴-۶-۲- مدیریت حافظه اصلی

قسمتی که در هر سیستم عامل وظیفه مدیریت حافظه را دارد به مدیر حافظه[2] معروف است و نحوه عملکرد این قسمت تاثیر فراوانی بر عملکرد کلی سیستم عامل دارد. در سیستم عامل های همه کاربره، مدیریت حافظه به تکنیک‌های مدرن مدیریت حافظه مجازی مشابه سیستم paging و swap هم مجهز است که در سیستم عامل‌های بلادرنگ و یا تعبیه شده از بسیاری از این تکنیک‌ها به جهت سادگی سیستم اصلا استفاده نمی شود.

۴-۶-۳- مدیریت فایل

مدیریت فایل یکی از مشهود ترین مؤلفه‌های سیستم عامل می باشد. کامپیوتر می تواند اطلاعات را در انواع مختلف رسانه‌های فیزیکی ذخیره کند و هر رسانه ویژگی‌های خاص خود را دارا ست. سیستم عامل‌های همه کاربره می باید دید منطقی یکنواختی را از فایل‌ها به کاربر و برنامه‌ها ارائه دهد در صورتی که در سیستم عامل‌های بلادرنگ و تعبیه شده، گاهی کاربر مستقیم و حتی دیسک سخت و سایر حافظه‌های ثانوی وجود ندارد و هیچ اطلاعاتی در قالب فایل‌ها نیست. در بسیاری از این سیستم عامل‌های تعبیه شده و بلادرنگ، احتیاجی به یک مدیریت مدرن برای فایل‌ها و دایرکتوری‌ها وجود ندارد و این سیستم عامل‌ها اطلاعات مورد

---

1 Earliest-Deadline–First Approach
2 Memory Manager

نیاز خود را در قالب دسترسی مستقیم به محدوده‌ای از حافظه فقط خواندنی (ROM) یا حافظه‌های متشابه استفاده می کنند.

۴-۶-۴- مدیریت سیستم ورودی / خروجی

سیستم‌عامل‌های همه کاربره معمولا ورودی‌ها و خروجی‌های سیستم کامپیوتر را به وسیله مدل‌های نرم‌افزاری بسیار ساده تری به کاربر و برنامه‌ها ارائه می کنند و اصولا به لحاظ امنیتی به کاربران و برنامه‌ها اجازه نمی دهند تا خود به طور مستقیم با این ورودی‌ها و خروجی‌ها ارتباط برقرار کنند؛ در صورتیکه در سیستم‌های بلادرنگ و یا تعبیه شده سیستم‌عامل به کمک برنامه‌ها می آید تا بتوانند به طور صحیح به ورودی‌ها و خروجی‌ها دسترسی داشته باشند و آنها را کنترل کنند. این اختلاف سبب می شود تا بخش مدیریت ورودی‌ها و خروجی‌ها و امنیت‌های لازم، در سیستم عامل‌های تعبیه شده بسیار ساده تر از سیستم‌عامل‌های همه کاربره باشد.

۴-۶-۵- مدیریت حافظه ثانوی

پردازنده اصلی تنها می‌تواند برنامه‌هایی که در حافظه اصلی وجودرا دارند اجرا کند و برنامه‌ها می توانند از اطلاعات خود که در حافظه اصلی هستند استفاده کنند. از آنجایی که حافظه اصلی توانایی ذخیره تمام برنامه‌ها و اطلاعات را ندارد و به لحاظ ساختار با قطع برق اطلاعات آن از بین خواهند رفت، حافظه های مجازی به وجود می آیند. سیستم های عامل همه کاربره در کنار سایر وظایف خود مدیریت حافظه ثانوی که معمولا دیسک سخت است را نیز بر عهده دارد؛ در حالیکه در بسیاری از سیستم عامل‌های بلادرنگ و تعبیه شده، دیسک سخت وجود ندارد و سیستم عامل اطلاعات دائمی خود را از حافظه‌های مانند ROM و EEPROM می خواند و برای ذخیره سازی اطلاعات میانی و تنظیمات می تواند از Flashها استفاده کند.

۴-۶-۶- شبکه

یکی از اجزای سیستم عامل که بخصوص در سیستم عامل‌های همه کاربره امروزی بسیار پر نقش می باشد مساله شبکه و مدیریت آن است. این در حالیست که در بسیاری از سیستم‌عامل‌های تعبیه شده و بلادرنگ، اصلا شبکه وجود ندارد. در برخی از این سیستم‌ها،

امکانات محدودی از شبکه (معمولا جهت ارتباطات و اطلاع رسانی) تعبیه شده است و در بعضی سیستم‌های تعبیه شده دیگر ، بر خلاف سایرین، مانند مسیریاب‌ها شبکه و سویچ‌ها، شبکه و حواشی آن، هدف اصلی سیستم عامل و برنامه‌های آن است و سیستم عامل امکانات بسیار فراوانی را جهت کنترل شبکه و تنظیم آن فراهم می آورد.

۴-۶-۷- سیستم حفاظتی

وجود چندین کاربر و یا اجرا چندین برنامه در کنار یکدیگر مساله حفاظت را به وجود می‌آورد. سیستم عامل های همه کاربره موظف هستند اطلاعات کاربران را از یکدیگر حفظ کند و حین اجرای برنامه ها، برنامه‌ها و اطلاعات آنها را از یکدیگر محفوظ نگه دارد. برای این امر الگوریتم‌ها نرم‌افزاری و مکانیزم‌های سخت‌افزاری مختلف برای کامپیوترهای چند کاربره و سیستم عامل های همه کاربره، تعبیه شده است. در سیستم عامل‌های بلادرنگ و یا تعبیه شده، صورت مساله بسیار متفاوت است. بسیاری از این سیستم‌ها اصلا در دسترس مستقیم و قابل تغییر کاربر نیستند(مانند کامپیوتر انژکتور اتومبیل) و چون برنامه‌های از قبل نوشته شده و ثابتی دارند، مساله امنیت در آنها به کلی فراموش می شود. برخی از سیستم عامل‌های خاص دیگر مانند سیستم عامل‌های مورد استفاده در سخت‌افزارهای دیواره‌های آتش، مساله امنیت در آنها به مراتب پیچیده‌تر از مساله امنیت در سیستم عامل‌های همه کاربره خواهد شد و اصولا این سیستم‌ها برای افزایش امنیت به وجود می آیند.

۴-۶-۸- مفسر فرمان یا پوسته

ارتباط سیستم‌عامل با کاربران توسط مفسر فرمان، پوسته یا متشابهات آن به وجود می‌آید. در بعضی از سیستم‌عامل‌های همه کاربره مانند MS-DOS و UNIX مفسر فرمان جدا از سیستم عامل است. بعضی از سیستم عامل‌ها، مانند ویندوز و مکینتاش، پوسته‌های مبتنی بر پنجره‌ها و ماوس را به کاربر ارائه می کنند که این پوسته‌ها به واسطه‌های دوستانه با کاربر[1] معروف شده اند.

---

1 User Friendly

در سیستم عامل‌های بلادرنگ و یا تعبیه شده، پوسته بسته به هدف سیستم به وجود می‌آید و سیستم‌عامل ممکن است فاقد یک پوسته قابل رویت باشد یا ظاهر گرافیکی بسیار مناسبی را به کاربر خود ارائه کند. مثلا سیستم‌عامل‌های تعبیه شده در تلویزیون‌های دیجیتال به کاربران خود امکان تنظیم پارامترهای صدا و تصویر را با کنترل از راه دور ارائه می کنند و معمولا از شکل و شمایل قابل قبول و ساده ای برخوردار هستند.

۴-۷- نتایج

در این قسمت مبانی سیستم عامل‌های بلادرنگ و تعبیه شده و نکات کلیدی آنها بررسی شد. با توجه به مطالب فوق، ویژگی‌های مطلوب برای یک سیستم عامل متن باز Enterprise بلادرنگ و یا تعبیه شده به شرح ذیل است:

۱- دارا بودن امکانات اختیاری و قابل انتخاب در سیستم‌عامل‌های کلاسیک (اشتراک زمانی، مدیریت پردازش ها، مدیریت حافظه، مدیریت فایل و...)

۲- امکانات توسعه و بهینه سازی سیستم‌عامل شامل افزودن ویژگی‌های خاص و یا کاهش خصوصیات سیستم عامل و نگارش راه اندازهای[1] جدید برای سخت‌افزارهای جانبی

۳- امکانات و الگوریتم های مناسب مدیریت پروسس‌های بلادرنگ در بخش مدیریت پروسس‌های سیستم عامل

۴- امکانات و الگوریتم‌های مناسب مدیریت حافظه اصلی و جانبی در سیستم‌های بلادرنگ و تعبیه شده

۵- امکانات و الگوریتم های مناسب مدیریت مستقیم ورودی‌ها و خروجی‌ها برای برنامه‌های تعبیه شده و یا بلادرنگ

۶- امکان انتقال سیستم عامل به سخت‌افزارهای مختلف مانند x86 ، RISC ، PowerPC و غیره.

۷- انطباق ساختارداخلی سیستم‌عامل با استانداردهایی نظیرPOSIX به‌جهت انتقال برنامه‌ها در لایه متن اولیه

---

1 Drivers

۸- امکانات امنیتی و وجود لایه های اختیاری امنیت داخلی و خارجی

بعضی از ویژگی‌های مطلوب جهت بومی سازی بطور مختصر عبارتند از:

۱- پایداری و خوش نامی سیستم عامل

۲- دارا بودن مجوز های متن باز لازم برای استفاده در ایران

۳- پشتیبانی گسترده و بروز توسط تیم‌ها یا شرکتهای معتبر

۴- پشتیبانی چند زبانی

۵- واسط کاربری قوی و حتی المقدور سازگار یا قابل انطباق جهت پشتیبانی زبان فارسی

فصل پنجم – بررسی و مقایسه سیستم عامل‌های بلادرنگ و نهفته متن باز موجود

امروزه تعداد کثیری از نرم‌افزارهای سیستم عامل‌های تعبیه شده و بلادرنگ متن باز وجود دارند که در این بخش سعی شده تا با بررسی مهم‌ترین عناوین این نرم‌افزارها به قابلیت‌های آنها اشاره شود و در حالت کلی با یکدیگر مقایسه شوند.

برای‌بررسی سیستم‌عامل‌ها ابتدا به تاریخچه، معرفی امکانات فنی و بررسی سیستم‌عامل‌های متن باز بلادرنگ و تعبیه شده پرداخته شده و سپس امکانات فنی و خصوصیات آنها با یکدیگر در جداول مختلف مقایسه و بررسی شده‌اند.

۵–۱– سیستم عامل ایده آل

اولین دلیل مشهود برای به وجود آمدن سیستم عامل‌های بلادرنگ و یا تعبیه شده، احساس نیاز به این نوع سیستم عامل‌ها است. طبیعی است اگر سیستم عامل‌های همه کاربره، جوابگوی همه مشکلات، نیازها و محدودیت‌ها بودند، هیچگاه سیستم عامل‌هایی با کاربردهای خاص به وجود نمی‌آمدند. اگر سیستم عاملی به‌عنوان یک سیستم عامل مناسب (سیستم عامل ایده‌آل) معرفی شود ولی احتیاجات و نیازها کلی نرم‌افزاری را برآورده نسازد و با محدودیت‌های مختلف منطبق نباشد، سیستم عامل مورد نظر بطور صحیح انتخاب نشده است.

نیازهای کلی سیستم عامل‌های بلادرنگ و سیستم عامل‌های تعبیه شده در فاز اول گفته شده است و در اینجا به شرایط، مشکلات و محدودیت‌هایی اشاره شده است که در انتخاب سیستم عامل ایده‌آل تاثیر دارند.

۵–۱–۱– محدودیت های سخت‌افزار

همه‌ی راهکارهای مبتنی بر کامپیوتر، از یک مجموعه سخت‌افزار مجتمع و یک مجموعه نرم‌افزار مجتمع استفاده می‌کنند. در بعضی راهکارهای سیستم‌های تعبیه شده و یا سیستم‌های بلادرنگ، به دلایل مختلف مانند دلایل اقتصادی، دلایل سیاسی و یا دلایل فنی لزوما می‌باید از سخت‌افزار مشخصی استفاده شود، برای مثال، عاقلانه‌ترین طریق برای بهبود عملکرد یک مجموعه و سیستم‌های موجود، غالبا بهینه ساختن نرم‌افزار است تا تغییر و تحول در سخت‌افزارهای موجود در آن مجموعه.

در صورت وجود محدودیت‌های سخت‌افزاری، نرم‌افزارها می باید خود را با سخت‌افزار خود تطبیق دهند و این وظیفه مهندس نرم‌افزار مربوطه است که بهترین طراحی را انجام دهد و بهترین گزینه‌ها را انتخاب کند. از سوی دیگر بسیاری از سخت‌افزارها فقط دارای مجموعه کوچکی از امکانات برای نرم‌افزار هستند. برای مثال میکرو کنترلرهای سری 8051 به علت محدودیت‌های سخت‌افزاری، تقریبا با هیچ سیستم عامل مدرنی سازگاری ندارند و بطور عمده در صنعت در سیستم‌های تعبیه شده استفاده می‌شوند. حتی اگر برنامه‌ای وظایف پایه‌ای سیستم عامل را در این میکروکنترلرها انجام دهد (مانند MON51)، این برنامه مطابق تعاریف ارائه‌شده در قسمت قبل، به سیستم عامل کاذب معروف است و قابل قیاس با سیستم عامل‌های مدرن امروزی نیست. درچنینی مواردی مشخص است که در یک پروژه خاص، قطعا بهترین انتخاب ممکن برای سیستم عامل با بررسی شرایط اولیه پروژه مشخص می‌شود و یک سیستم عامل ایده‌آل در همه موارد نمی تواند جوابگوی همه محدودیت‌های سخت‌افزاری باشد.

۵-۱-۲- محدودیت های نرم‌افزار

در بعضی مساله‌ها و پروژه‌ها، نرم‌افزارها هستند که شرایط کاری را مشخص کرده و محدودیت‌هایی را اعمال می‌کنند. برای مثال فرض کنید که هدف یک پروژه، تولید یک سیستم صنعتی و تعبیه شده برای مانیتورینگ یک دستگاه مکانیکی بزرگ باشد و برنامه‌های نرم‌افزاری لازم به زبان C در MS-DOS قبلا نگارش شده باشند و امکان تغییر و یا انتقال آنها وجود نداشته باشد. در چنین حالتی سیستم عامل و سخت‌افزار تعبیه شده می‌باید در قدم اول، اهداف پایه‌ای نرم‌افزار را بطور مطلوب برآورده سازد و مدیر نرم‌افزار، امکان انتخاب گسترده‌ای نخواهد داشت.

به‌عنوان یک مثال دیگر، تصور کنید که در یک پروژه هدف این باشد که با استفاده از امکانات برنامه SAMBA، یک دستگاه سرور چاپگر به صورت تعبیه شده، تولید شود. در این مثال، کارشناس پروژه می‌باید در انتخاب سیستم‌عامل و سخت‌افزار آن به گونه‌ای رفتار نماید که برنامه SAMBA و ابزارهای مرتبط بتوانند به صورت بهینه عمل کنند.

نتیجه بحث فوق به این صورت است که نوع برنامه‌ها و کاربرد آنها در انتخاب سیستم عامل تاثیر دارد و یک سیستم عامل کلی و ایده‌آل را نمی‌توان جوابگوی همه نرم‌افزارها دانست.

۵-۱-۳- محدودیت های مجوزها

لیسانس یا مجوز استفاده از برنامه مختلف، دیر یا زود در ایران مشابه تمام دنیا، به مساله‌ای بزرگ تبدیل خواهد شد. با وجود اینکه در ایران هنوز قانونی برای منع استفاده از نرم‌افزارهای بدون مجوز خارجی وجود ندارد، ولیکن از همین امروز می‌باید برای فرداها اندیشید.

از سوی دیگر استفاده از نرم‌افزارهای متن باز مانند تیغ دو لبه است. اگر در استفاده از نرم‌افزارهای متن باز به لیسانس‌ها و مجوزها، دقت شود و موارد و بندهای این مجوزها به طور دقیق رعایت شوند، هیچ ضرری متوجه کسی نخواهد بود. در حالیکه در صورت توجه نکردن به مجوزهای برنامه‌های متن باز، برای مثال استفاده کردن و تغییر دادن کد دارای مجوز GPL و منتشر نکردن کدهای اولیه منبع، لبه خطرناک این مجوزهاست و ممکن است سبب مشکلات قانونی و مسائل دیگر شود.

در کل، اگر در یک پروژه لازم است که کد منبع فاش نشود، نمی‌توان در آن پروژه از کدهای منبع با مجوز GPL استفاده کرد و در صورت درخواست خریدار یا ارائه و فروش نتایج پروژه در مجامع عمومی، کد منبع باید در محلی که قابل دسترس همگان باشد، قرار داده شود. این در حالی است که مجوز BSD آزادی عمل بسیار زیادی را به نویسنده در مخفی نگاه داشتن کد نهایی می‌دهد.

با توجه به اینکه در حال حاضر هیچ سیستم عاملی وجود ندارد که جوابگوی همه لیسانس‌ها و کاربردها باشد، پس نمی‌توان یک سیستم عامل ایده‌آل را برای همه کاربردها در نظر گرفت.

۵-۱-۴- نتیجه گیری از سیستم عامل ایده آل

با توجه به موارد فوق، هیچ سیستم عامل بلادرنگ یا تعبیه شده‌ای جوابگوی همه نیازها نخواهد بود و هیچ سیستم عاملی را نمی‌توان یافت که با همه محدودیت‌های سخت‌افزاری و نرم‌افزاری سازگار باشد و در نتیجه سیستم عامل ایده‌آل در این زمینه وجود نخواهد داشت. در این قسمت هدف این نیست که یک سیستم عامل ایده‌آل معرفی شود، بلکه این است که مهم‌ترین عناوین در سیستم عامل‌های بلادرنگ و تعبیه شده بررسی شوند و یک شناخت عمومی و مقایسه کلی انجام گیرد تا در آینده کارشناسان و خوانندگان این قسمت بتوانند در هر پروژه سیستم عامل خود را به درستی انتخاب کنند.

۵-۲- ساختار داخلی سیستم عامل ها

کلاً سیستم عامل‌های متن باز عملیاتی منجمله سیستم عامل‌های بلادرنگ و تعبیه شده را می‌توان در یک تقسیم‌بندی با توجه ساختار داخلی (API) به دو دسته ۱- سیستم عامل‌های منطبق بر استاندارد POSIX و ۲- سیستم عامل‌های غیرمنطبق بر استاندارد POSIX تقسیم کرد که در زیر به‌طور مختصر به آنها اشاره شده است.

۵-۲-۱- سیستم عامل های عملیاتی متشابه با یونیکس

مهم‌ترین و معروف‌ترین سیستم عامل‌های همه کاربره‌ای که در این رده جای دارند عبارتند از : OpenSolaris، خانواده BSD شامل FreeBSD ، OpenBSD و NetBSD ، لینوکس و OpenDarwin. این سیستم عامل‌ها سیستم عامل‌های پرمصرف و کاربردی هستند که کد منبع آنها باز است و اکثرا با استاندارد POSIX منطبق هستند. ازمیان سیستم عامل‌های عملیاتی متشابه با یونیکس ، از سیستم عامل لینوکس و خانواده BSD به وفور در سیستم‌های تعبیه شده و سیستم‌های بلادرنگ نیز استفاده می‌شود.

۵-۲-۲- سیستم عامل های عملیاتی غیر متشابه با یونیکس

مهم‌ترین و معروف‌ترین سیستم عاملی که در این رده جا دارد و در سیستم‌های بلادرنگ و تعبیه شده نیز استفاده می‌شود، سیستم عامل FreeDOS است که می‌توان آن را نسخه متن‌باز سیستم عامل MS-DOS نامید که در ادامه گزارش بطور مفصل معرفی و بررسی شده است.

۵-۲-۳-نتیجه گیری ساختار داخلی

بهتر است در پروژه‌های جدید و پروژه‌هایی که کارشناسان نرم‌افزاری آن پروژه قدرت انتخاب از میان سیستم عامل‌ها را دارند، از سیستم عامل‌های منطبق با استاندارد POSIX استفاده کنند. متشابه بودن سیستم عامل انتخابی با یونیکس و منطبق بودن آن با استانداردهایی چون POSIX این امکان را می‌دهد که از بسیاری نرم‌افزارهای متن باز دیگر که از این استاندارد حمایت می‌کنند در پروژه استفاده شود و در لایه کد اولیه، به سیستم عامل مورد نظر منتقل شوند.

۵-۳- بررسی سیستم عامل های تعبیه شده و بلادرنگ

در زیر مهم‌ترین سیستم عامل‌های تعبیه شده و بلادرنگ معروف و پرمصرف متن‌باز و عملیاتی امروزی به طور مختصر بررسی شده است. دقت شود که بعضی از این سیستم عامل‌ها در کاربردهای عمومی هم استفاده دارند و بخاطر خوشنامی و امکانات بالقوه، در سیستم‌های تعبیه شده یا بلادرنگ نیز استفاده می‌شوند.

۵-۳-۱- سیستم عامل لینوکس

۵-۳-۱-۱ - لینوکس در نقش سیستم عامل تعبیه شده

کد منبع هسته سیستم عامل لینوکس از طریق سایت Kernel.org قابل دسترسی است و برنامه نویسان می‌توانند به راحتی این کد را به همراه تمام راه‌اندازهای سخت‌افزارهای موجود دریافت کرده و در پروژه‌های سیستم‌های تعبیه شده خود استفاده کنند. این سبب شده است تا استقبال فراوانی از لینوکس در دنیای تعبیه شده به وجود آید.

اصولا کد هسته لینوکس بسیار پایدار است و می توان آن را با سخت‌افزارهای موجود به راحتی منطبق کرد و نسخه تعبیه شده این کد را به وجود آورد . بسیاری از شرکت‌های معروف در زمینه سیستم عامل‌های تعبیه شده مانند VxWorks ، که سال‌ها از کد خصوصی خود استفاده می‌کرده‌اند، امروزه سرویس های تعبیه شده منطبق با کد هسته لینوکس را در آدرس http://www.windriver.com/linux به مشتریان خود ارائه می‌کنند.

۵-۳-۱-۲ -لینوکس در نقش سیستم عامل بلادرنگ

هسته خالص سیستم عامل لینوکس که به Vanilla Linux معروف است، تا نسخه 2.6 بدون patchهای اضافی قدرت لازم را برای سیستم‌های بلادرنگ ندارد و در نسخه 2.6 هسته، امکاناتی نظیر CONFIG_PREEMPT به آن اضافه شده‌است که امکانات نرم‌افزاری بلادرنگ را در هسته بهبود می‌بخشد و این قدرت را به هسته لینوکس می‌دهد تا با درصدی معادل ۹۹.۹٪ به سیستم‌های بلادرنگ، امکانات لازم را ارائه کند. این درصد، برای سیستم‌های بلادرنگ نرم و بسیاری از راهکارها، مناسب و کافی است. پروژه‌های افراد ثالث در کنار امکانات هسته لینوکس وجود دارد که با ارائه patchهای لازم، هسته لینوکس را حتی برای سیستم‌های بلادرنگ سخت هم مناسب می‌کند.

در حال حاضر فعالیت زیادی برای افزودن امکانات بلادرنگ در کد منبع هسته لینوکس توسط تیم اصلی توسعه هسته در حال انجام است که به مرور زمان در کد اصلی هسته نیز اضافه خواهند شد.

۵-۳-۱-۳ - سخت‌افزارهای پشتیبانی شده درلینوکس

لینوکس تقریبا از همه سخت‌افزارها و پردازنده‌های متداول پشتیبانی می‌کند و تعداد این سخت‌افزارها همه روزه در حال افزایش است. برای دیدن لیست به روز سخت‌افزارها می توانید به سایت linuxhardware.org مراجعه کنید.

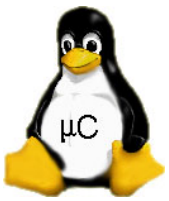
لیست زیر نمونه‌ای از این سخت‌افزارهایی است که توزیع کننده Debian از آنها پشتیبانی می‌کند. برای دیدن لیست به‌روز سخت‌افزارها می‌توانید به‌سایت debian.org/portsمراجعه کنید.

Intel x86,IA-32
Motorola 68k
Sun SPARC
Alpha
Motorola/IBM PowerPC
ARM
MIPS CPUs
HP PA-RISC
IA-64
S/390

۵-۳-۲- پروژه سیستم عامل ucLinux

۵-۳-۲-۱ - تاریخچه سیستم عامل ucLinux

پروژه ucLinux بصورت "You See Linux" تلفظ می‌شود و در ابتدا یک انشقاق از هسته نسخه ۲.۰ سیستم‌عامل لینوکس بود. این پروژه در اصل برای پردازنده‌ها و کنترلرهایی که سیستم سخت‌افزاری مدیریت حافظه(MMU) ندارند، بوجود آمده است. این پروژه جزء یکی از بزرگترین پروژه‌های سیستم‌های تعبیه شده محسوب می‌شود و بسیاری از سخت‌افزارهای موجود و میکروکنترلرهای فعلی از آن به خوبی حمایت می‌کنند.



شکل ۵-۱: آرم سیستم عامل ucLinux

این پروژه همه روزه در حال رشد است و امروزه هسته‌های جدیدتر لینوکس (۲.۴ و ۲.۶) نیز در این پروژه استفاده می‌شوند. این پروژه به صورت یک هسته مرکزی و مجموعه نسبتا کاملی از ابزارهای کاربردی و ابزارهای لازم برای توسعه عرضه می‌شود.

۵-۳-۲-۲ - سخت‌افزارهای پشتیبانی شده در سیستم عامل ucLinux

لیست زیر نمونه‌ای از سخت‌افزارهایی است که این پروژه می‌شناسند و از آنها پشتیبانی می‌کند:

- پردازنده‌های Motorola DragonBall[1] و سری 68K که از آن منشق شده‌اند.
- پردازنده‌های Motorola ColdFire
- پردازنده‌های ADI Blackfin[2] محصول Analog Devices که در وب سایت این شرکت نیز سیستم عامل ucLinux به عنوان یک سیستم عامل متن باز معرفی و لیست شده‌است[3]. این پردازنده برای سیستم‌هایی که یک به DSP در کنار پردازنده اصلی احتیاج دارند مانند دوربین‌های دیجیتال و کاربردهای دیگر صوتی/ تصویری ایده‌آل به نظر می‌رسد.
- پردازنده های ETRAX که محصول شرکت AXIS هستند. شرکت AXIS در وب سایت خود تبلیغ سیستم عامل متن باز لینوکس را در کنار پردازنده‌های خود می‌کند و محصولاتی مشابه دوربین‌های تحت شبکه، سرورهای چاپگر و محصولاتی دیگر را به کمک لینوکس و سخت‌افزار خود تولید کرده است.
- پردازنده‌های motorola QUICC

---

1 http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=MC68360&node
2 http://www.analog.com/processors/blackfin/
3 http://www.analog.com/processors/blackfin/gettingStarted/operatingSystems/index.html

- پردازنده‌های ARM7TDMI و MC68EN302
- سری پردازنده‌های Sigma Designs که عمدتا برای پردازش‌های صوتی و تصویری استفاده می‌شوند و از سیستم عامل لینوکس در محصولات این شرکت به خوبی استفاده می‌شود و این شرکت حتی کد متن برنامه‌های خود را نیز منتشر کرده است.
- پردازنده‌های i960 محصول شرکت Intel. این پردازنده‌ها از خانواده RISC هستند و برای کاربردهای شبکه و بسیاری از کاربردهای عمومی مناسبند.
- پردازنده‌های ISICAD Prisma که بخوبی از ucLinux، Linux و Minix پشتیبانی می‌کنند.
- پردازنده‌های Atari 68k
- پردازنده‌های Microblaze که یک پردازنده قابل پیاده‌سازی در Xilinx FPGA است. این پردازنده یک پردازنده از خانواده RISC است و چون دست طراح در طرز پیاده‌سازی این سخت‌افزار در FPGA باز است، در صنعت بسیار حائز اهمیت است.
- پردازنده‌های V850E محصول شرکت NEC[1] که از خانواده RISC هستند.
- پردازنده‌های H8 محصول شرکت Hitachi[2]

در پروژه ucLinux چون از واحد MMU در پردازنده اصلی (حتی در صورت وجود) صرف‌نظر می شود، سرعت سیستم بالا میرود ولی در سیستم عامل ucLinux تکنیک حفاظت از اطلاعات در حافظه و تکنیک حافظه مجازی وجود ندارد. در این حالت اگر یک برنامه مختل شود، می‌تواند در کل سیستم، حتی در فضای هسته سیستم عامل، موثر باشد.

۵-۳-۳- سیستم عامل FreeBSD

۵-۳-۳-۱ - سیستم عامل FreeBSD در نقش سیستم عامل تعبیه شده

سیستم‌عامل FreeBSD با داشتن کد منبع باز و لیسانس مناسب به بستر بسیاری از پروژه‌های تعبیه شده، تبدیل شده است. بسیاری از محصولات تعبیه شده بطور مستقیم‌یا

1 http://www.necel.com
2 http://www.renesas.com

غیرمستقیم از FreeBSD استفاده کرده‌اند. نمونه‌ای از این محصولات: Juniper Router، Nokia's Firewall و سیستم عامل بلادرنگ و تعبیه شده معروف خصوصی VxWorks است.

۵-۳-۳-۲ - سیستم عامل FreeBSD در نقش سیستم عامل بلادرنگ

با توجه به اینکه FreeBSD نسبتا از استاندارد POSIX و استانداردهای BSD به خوبی حمایت می‌کند، از سیستم عامل FreeBSD می‌توان در سیستم‌های بلادرنگ (بخصوص سیستم‌های بلادرنگ نرم) که چندین نرم‌افزار در اولویت‌های مختلف می‌باید اجرا شوند، استفاده کرد.

در جدول(۵-۱) اولویت‌های موجود در هسته سیستم عامل FreeBSD آورده شده است. همانطور که مشاهده می‌شود سیستم عامل FreeBSD از ۵ اولویت مختلف پشتیبانی می‌کند[1].

جدول۵-۱: اولویت‌های موجود در هسته سیستم عامل FreeBSD

	کلاس زمانبندی
۰ الی ۶۳	وقفه‌ها
۶۴ الی ۱۲۷	نیمه بالای هسته
۱۲۸ الی ۱۵۹	سیستم‌های بلادرنگ کاربر
۱۶۰ الی ۲۲۳	کاربر عادی با سیستم اشتراک زمانی
۲۲۴ الی ۲۵۵	کاربر بی‌کار (IDLE)

۵-۳-۳-۳ - سخت‌افزارهای پشتیبانی شده در سیستم عامل FreeBSD

سیستم عامل FreeBSD مطابق اطلاعیه موجود در وب سایت این سیستم عامل، از مجموعه سخت‌افزارهای زیر حمایت می‌کند : IA-32 ، UltraSPARC ، IA-64 ، PC-98 و ARM. لیست به روزتر این سخت‌افزارها در وب سایت این شرکت FreeBSD.org موجود است.

1 http://www.opensolaris.org

۵-۳-۴- سیستم عامل NetBSD

۵-۳-۴-۱ - سیستم عامل NetBSD در نقش سیستم عامل تعبیه شده

سیستم عامل NetBSD با پشتیبانی از بسیاری از سخت‌افزارها، به یکی از محبوب‌ترین سیستم عامل‌ها در دنیای سیستم عامل‌های تعبیه شده تبدیل شده است. مجموعه‌ای از توضیحات مختصر و مفید و لینک‌هایی در زمینه سیستم‌های تعبیه شده در وب سایت این سیستم عامل در آدرس http://www.netbsd.org/Misc/embed.html وجود دارد. توجه کنید که اگر در یک پروژه از یک‌سو سخت‌افزار مورد نظر از نوع PowerPC یا MIPS و یا سخت‌افزارهای دیگری از خانواده BSD باشد، فقط سیستم عامل NetBSD از آنها حمایت می‌کند و از سوی دیگر کد نهایی پروژه لزوماً می‌باید به صورت متن بسته و خصوصی باقی بماند، تقریباً تنها گزینه ممکن سیستم عاملNetBSD خواهد بود.

۵-۳-۴-۲ - سخت‌افزارهای پشتیبانی شده در سیستم عامل NetBSD

لیست کامل سخت‌افزارها و پردازنده‌های پشتیبانی شده در سیستم عامل NetBSD آنقدر مفصل است که در این بخش آورده نشده است. برای دریافت لیست کامل می توانید به آدرس http://www.netbsd.org/Ports مراجعه فرمایید.

۵-۳-۵- سیستم عامل OpenBSD

۵-۳-۵-۱- امنیت در سیستم عامل OpenBSD

گرچه سیستم عامل OpenBSD در محافل مختلف به سیستم عاملی بسیار امن توصیف می‌شود، ولی این سیستم عامل تنها از تکنیک عمومی اجازه دسترسی به فایل‌ها که تقریبا در تمام نگارش‌های یونیکس وجود دارد برای کنترل منابع خود استفاده می‌کند و در عوض تمرکز بیشتری در بهینه‌سازی هسته و رفع اشکالات امنیتی در برنامه‌ها دارد.

۵-۳-۵-۲- سیستم عامل OpenBSD در نقش سیستم عامل های تعبیه شده

سیستم عامل OpenBSD همانند سایر اعضای خانواده BSD بطور عمده در تولید سیستم‌عامل‌های تعبیه شده استفاده می‌شود. از سیستم عامل OpenBSD بیشتر برای سیستم‌های تعبیه شده در شبکه‌ها که نقش Router و Firewall را دارند استفاده می‌شود.

نمونه‌ای از این سیستم‌ها پروژه opensoekris.sourceforge.net است که مانند یک Router Cisco عمل می‌کند و متن آن از سیستم عامل OpenBSD اقتباس شده است. از سیستم عامل OpenBSD همچنین در PDAها نیز بطور گسترده استفاده شده است.

۵-۳-۵-۳- سخت‌افزارهای پشتیبانی شده در سیستم عامل OpenBSD

مشابه سیستم عامل NetBSD، سیستم عامل OpenBSD از سخت‌افزارهای متعددی پشتیبانی می‌کند که لیست کامل آنها در آدرس http://www.openbsd.org/plat.html وجود دارد.

۵-۳-۶- سیستم عامل FreeDOS

۵-۳-۶-۱- سیستم عامل FreeDOS در نقش سیستم عامل تعبیه شده و بلادرنگ

با توجه به اینکه MS-DOS یک سیستم عامل چند کاربری و اشتراک زمانی نبوده و مسائل و الگوریتم‌های امنیتی امروزه سیستم عامل‌ها در آن وجود نداشته است، سیستم عامل FreeDOS نیز به همین صورت طراحی شده است و در این سیستم عامل هر برنامه‌ای که اجرا می‌شود، تقریبا تمام قدرت سیستم عامل و کامپیوتر را در اختیار خود می‌گیرد. سادگی سیستم عامل FreeDOS برای برنامه‌های تعبیه شده و بلادرنگ که مساله اشتراک زمانی و کنترل‌های سیستم عامل های امروزی گاهی به ضرر آنها ختم می‌شود، بسیار مناسب است و در عوض عمده مسئولیت‌ها به دوش خود برنامه‌ها خواهد بود.

۵-۳-۷- سیستم عامل eCos

۵-۳-۷-۱- تاریخچه سیستم عامل eCos

سیستم عاملeCos یک سیستم عامل بسیار سبک، متن باز و کاملا رایگان برای سیستم‌های بلادرنگ و تعبیه شده است. نام این پروژه از سرواژه‌های Embedded Configurable Operating System تشکیل شده است. این پروژه شرایط بسیار مناسبی برای تنظیم سیستم عامل با نیازهای برنامه‌ها فراهم کرده است تا کمترین سربار را در سیستم به وجود آورد. در سخت‌افزارهایی که به علت محدودیت های مختلف نتوان از لینوکس یا سیستم عامل‌های مدرن دیگر استفاده کرد، سیستم عامل eCos گزینه مناسبی خواهد بود.

شکل ۵-۲: آرم سیستم عامل eCoc

پروژه eCos ابتدا در Cygnus Solution طراحی و توسعه پیدا کرد که در نهایت شرکت Red Hat آن را خرید. در اوایل سال ۲۰۰۲ شرکت Red Hat فعالیت خود را بر روی این پروژه متوقف کرد و اعضای این پروژه با شرکت Red Hat شرکت دیگری به نام eCosCentric تشکیل دادند تا بتوانند فعالیت تجاری خود را در این پروژه ادامه دهند. در ابتدای سال ۲۰۰۴ با توجه به درخواست تیم توسعه دهنده پروژه eCos، شرکت Red Hat موافقت کرد تا لیسانس این پروژه را به سازمان نرم‌افزارهای متن باز (FSF) منتقل کند و این پروسه در اواخر سال ۲۰۰۵ اجرا شد.

سیستم عامل eCos کتابخانه‌ها و محیط‌های توسعه بسیار مناسبی را برای سیستم‌های تعبیه شده و بلادرنگ دربر دارد.

۵-۳-۷-۲- امنیت در سیستم عامل eCos

سیستم عامل eCos برای کاربردهای همه منظوره طراحی نشده است که مشابه لینوکس یا FreeBSD از امکانات امنیتی داخلی قدرتمندی استفاده کند و هدف اصلی آن بهینه بودن است. با این وجود، مساله امنیت در لابه‌لای مفاهیم این سیستم عامل، برای مثال در کتابخانه‌های SNMP و Web این سیستم عامل به چشم می‌خورد.

۵-۳-۷-۳- فایل سیستم در سیستم عامل eCos

سیستم عامل eCos از فایل سیستم‌های معمول در Linux و FreeBSD حمایت نمی‌کند. توجه این سیستم عامل به محدودیت سخت‌افزارهاست و تنها از فایل سیستم‌های ROM ، RAM ، JFFS2 و FAT که مناسب سیستم‌های تعبیه شده هستند، حمایت می‌کند.

۵-۳-۷-۴- سخت‌افزارهای پشتیبانی شده در سیستم عامل eCos

سیستم عامل eCos از سخت‌افزارهای متنوعی شامل پروسسورها و میکروکنترلرهای ۱۶ بیتی، ۳۲ بیتی و ۶۴ بیتی را پشتیبانی می‌کند که لیست زیر نمونه‌ای از این سخت‌افزارها است.

ARM

CalmRISC
FR-V
Hitachi H8
IA-32
Motorola 68000
Matsushita AM3x
MIPS
NEC V8xx
PowerPC
SPARC
SuperH
Nios II

۵-۳-۸- سیستم عامل FreeRTOS

۵-۳-۸-۱- بررسی سیستم عامل FreeRTOS

سیستم عامل FreeRTOS یک هسته بسیار سبک، متن باز و کاملا رایگان برای سیستم‌های بلادرنگ و تعبیه شده است. هسته این سیستم عامل به زبان C نگارش شده و به بسیاری از بسترهای سخت‌افزاری که قدرت اجرای یک سیستم عامل بزرگ را ندارند، حمل شده است. این سیستم عامل دارای لیسانس GPL با یک بند اضافی خاص است که امکان استفاده از این سیستم عامل را در پروژه‌های متن بسته می‌دهد.

شکل ۵-۳: آرم سیستم عامل free RTOS

این سیستم عامل اولین سیستم عامل متن باز و بلادرنگ و تعبیه شده کاربردی برای میکروکنترلرهای CORTEX-M3 بوده است. ابزارهای معرفی شده برای توسعه در این پروژه اکثرا تحت ویندوز هستند (اگرچه معمولا نسخه تحت لینوکس هم دارند) و در مثال‌های این پروژه از سیستم عامل ویندوز به‌عنوان محیط مادر برای نگارش برنامه‌های تعبیه شده استفاده شده است.

اهمیت این سیستم عامل در پشتیبانی آن از سخت‌افزارهایی مانند 8051 است که تقریبا هیچ سیستم عامل مدرنی آنها را پشتیبانی نمی‌کنند و قابلیت‌های بلادرنگی را به برنامه‌های این سیستم عامل‌ها می‌دهد.

۵-۳-۸-۲- سخت‌افزارهای پشتیبانی شده در سیستم عامل FreeRTOS

لیست زیر سخت‌افزارهایی هستند که این سیستم عامل از آنها پشتیبانی می‌کند:

ARM Cortex-M3
ARM architecture ARM7
AVR
x86
PIC microcontroller PIC18
Renesas H8/S
MSP430
HCS12
8052
MicroBlaze

۵-۳-۹- سیستم عامل RTLinux

۵-۳-۹-۱- بررسی سیستم عامل RTLinux

هسته سیستم عامل RTLinux ترکیب یک هسته بلادرنگ (هسته اصلی) به همراه یک هسته دیگر که معمولا هسته سیستم عامل Linux یا یکی از هسته‌های خانواده BSD است، می‌باشد. این سیستم عامل در اصل توسط Yodaiken، در New Mexico Institute of Mining and Technology نگارش شده است.

شکل ۵-۴: آرم سیستم عامل RTLinux

در سیستم عامل RTLinux تمام کنترل کامپیوتر شامل وقفه‌ها، دستگاه‌های ورودی و خروجی، پردازنده اصلی و غیره، دردست هسته اصلی (هسته بلادرنگ) می‌باشد و با رخ دادن وقفه‌ها، هسته اصلی کنترل کامپیوتر را در دست گرفته و برنامه‌های بلادرنگ را بدون واسطه (بدون دخالت هسته ثانوی) و در کمترین زمان ممکن اجرا می‌کند. هسته ثانوی که معمولاً هسته سیستم عامل لینوکس است با اولویت بسیار پایین، مانند یک برنامه بدون اولویت در هسته اصلی اجرا می‌شود. برای اینکه هسته ثانوی بتواند به‌درستی عمل کند، هسته اصلی در

صورت لزوم، رخ دادن وقفه‌ها را به هسته ثانوی اطلاع داده و آنها را برای هسته ثانوی شبیه‌سازی نرم‌افزاری می‌کند.

با ترکیب دو هسته، هم می‌توان در این سیستم عامل از امکانات هسته بلادرنگ استفاده کرد و هم می‌توان از امکانات یک هسته عمومی برای بسیاری کاربردهای غیر بلادرنگ سود جست. برنامه‌هایی که برای هسته اصلی این سیستم عامل نگارش می‌شوند مانند ماژول‌های لینوکس هستند و از استانداردهای متداول آنچنان پیروی نمی‌کنند. قدرت هسته اصلی و سرعت پاسخ دهی این سیستم عامل حتی برای سیستم‌های بلادرنگ نوع سخت به خوبی کافی است.

هسته غیر بلادرنگ و ثانوی در سیستم عامل RTLinux که معمولا هسته لینوکس است، یک سیستم عامل کاملا مستقل برای کاربران و برنامه‌ها به وجود می‌آورد. این هسته به نوعی از عملکرد و وجود هسته اصلی بی‌اطلاع است و کاملا مستقل از آن عمل می‌کند. این هسته مشابه همه هسته‌های لینوکس دیگر است و قدرت اجرای تمام برنامه‌های لینوکس را دارد.

مزیت اصلی سیستم عامل RTLinux وجود دو هسته بلادرنگ و هسته همه کاربره در کنار یکدیگر است که اجازه می‌دهد مجموعه برنامه‌های بلادرنگ و برنامه‌های عادی و کاربردی در کنار یکدیگر در یک کامپیوتر اجرا شوند.

اگرچه سیستم عامل RTLinux با لیسانس GPL توزیع می‌شود ولی مجموعه FSM Labs با اخذ هزینه، از سیستم عامل RTLinux نیز پشتیبانی می‌کند و ابزارها و محیط‌های توسعه مخصوص برای این منظور نگارش کرده است و نسخه RTLinux که دارای هسته BSD است فقط در قبال پرداخت هزینه، قابل استفاده است.

۵-۳-۹-۲- سخت‌افزارهای پشتیبانی شده در سیستم عامل RTLinux

چون در سیستم عامل RTLinux کنترل و محاوره اصلی با سخت‌افزار در دست هسته بلادرنگ است، سخت‌افزارهای پشتیبانی شده در این سیستم نسبت‌به هسته لینوکس معمول بسیار کاهش می‌یابد. نسخه رایگان این سیستم‌عامل در حال حاضر فقط از سخت‌افزارهای x86 و پروسسورهای قدیمی‌تر ARM حمایت می‌کند.

متاسفانه مجموعه FSM Labs امکانات موجود در نسخه تجاری این سیستم عامل به‌نام RTLinuxPro را در نسخه رایگان این سیستم عامل ارائه نکرده است. در نسخه تجاری این

سیستم عامل، هم از مجموعه وسیع‌تری از سخت‌افزارها پشتیبانی شده است و هم از هسته FreeBSD و NetBSD می‌توان به‌عنوان هسته همه کاربره استفاده کرد و از هسته لینوکس 2.6 هم به‌خوبی پشتیبانی شده است در حالیکه در نسخه رایگان فقط از امکانات هسته لینوکس 2.4 می‌توان استفاده کرد.

۵-۳-۱۰- سیستم عامل Minix

۵-۳-۱۰-۱- تاریخچه سیستم عامل Minix

همانطور که قبلا ذکر شد، سیستم عامل Minix نگارش پروفسور اندرو تننبام است و این سیستم عامل با هدف تدریس در دانشگاه‌ها نگارش شده است.

شکل ۵-۵: آرم سیستم عامل Minix

به وجود آمدن سیستم عامل Minix تاریخچه جالبی دارد. کد منبع سیستم عامل UNIX تا نسخه ۶ با لیسانس AT&T به دانشگاه‌ها ارائه می‌شد و AT&T اجازه داده بود تا از کد منبع این سیستم‌عامل در دانشگاه‌ها برای مسائل آکادمیک استفاده شود. در همین زمان، در بسیاری از دانشگاه‌ها سیستم عامل UNIX در درس سیستم‌عامل تدریس می‌شده است. در نسخه ۷ این سیستم عامل، AT&T کد منبع این سیستم عامل را دیگر به دانشگاه‌ها ارائه نکرد و این موضوع سبب شد تا بسیاری از دانشگاه‌ها درس سیستم عامل را به صورت تئوری تدریس کنند. آقای تننبام شخصا اعتقاد داشت که این درس را نمی‌توان به صورت تئوری تدریس کرد و سیستم عامل Minix را با هدف تدریس در دانشگاه‌ها نگارش کرد و کتاب Operating Systems: Design and Implementation خود را در زمینه سیستم عامل‌ها و Minix به چاپ رسانید. نسخه اولیه این سیستم عامل حدوداً ۱۲۰۰۰ خط کد منبع بود و در سال ۱۹۸۷ ارائه شد. اگرچه لیسانس اولیه سیستم عامل Minix از توزیع رایگان این سیستم

عامل جلوگیری می‌کرد ولی کد منبع آن راه را برای بسیاری از سیستم عامل‌ها منجمله لینوکس هموار کرد.

در سال۲۰۰۰ کد منبع Minix با مجوز BSD ارائه شد و توزیع این سیستم عامل و فعالیت‌های مختلف در زمینه این سیستم عامل به طور وسیع آغاز گشت. در حال حاضر نسخه ۳.۲.۱ این سیستم عامل موجود است. نسخه ۳ این سیستم عامل با نسخه‌های قبلی اختلافات وسیعی در ساختار داخلی هسته و سیستم عامل دارد. نسخه ۳ این سیستم عامل با هسته Microkernel ارائه شده است و هسته این سیستم عامل فقط در حدود ۴۰۰۰ خط کد دارد و سرویس های مختلف هسته در فضا و مد کاربر (و نه مد سیستم) اجرا می شود. این سبب سادگی این سیستم عامل خواهد شد و اگر یک سرویس خاص یا یک راه انداز سخت‌افزاری از کار بیفتد، به جای اینکه کل سیستم عامل از بین برود، هسته مرکزی مجددا سرویس را راه اندازی می کند.

به دلایل فوق سیستم عامل Minix نگارش ۳ بسیار پایدار است و بخاطر پایداری، سادگی کد منبع و مجوز BSD خود، برای سیستم‌های تعبیه شده محیط مناسبی است.

۵-۳-۱۰-۲- سخت‌افزارهای پشتیبانی شده در سیستم عامل Minix

نسخه ۳ سیستم عامل Minix در حال‌حاضر فقط در سیستم‌های x86 به‌خوبی کار می‌کند و انتقال این سیستم عامل به سخت‌افزارهای مختلف در حال انجام است.

نسخه‌های قبلی این سیستم عامل غیر از محیط x86 به سخت‌افزارهای Amiga ، Atari ، Macintosh و Sparc هم انتقال یافته‌اند و قابل استفاده هستند.

۵-۳-۱۰-۳- سیستم عامل Minix به عنوان سیستم عامل بلادرنگ و تعبیه شده

همانطور که در بخش معرفی این سیستم عامل ذکر شد، سیستم عامل Minix برای کاربردهای تعبیه شده بسیار مناسب است ولی در کاربردهای بلادرنگ، هنوز کد اصلی این سیستم عامل از امکانات نرم‌افزاری لازم حمایت نمی‌کند.

در حال حاضر پروژه‌هایی نظیر RT-Minix و Minix4RT برای به‌وجود آوردن امکانات بلادرنگ لازم در این سیستم عامل در حال اجرا است.

۵-۴ – مقایسه کاربردی

در جدول (۵-۲)با توجه به کاربرد سیستم عامل‌ها دسته‌بندی شده‌اند:

جدول ۵-۲: جدول مقایسه کاربردی سیستم عامل ها

Embedded	Linux , Minix, ucLinux
Real-time & Embedded	eCos , RTLinux, FreeRTOS, FreeBSD*, NetBSD*, OpenBSD*, FreeDOS**

*: فقط مناسب برای بلادرنگ نرم **: عملکرد بلادرنگ، اشتراک زمانی و اولویت‌بندی در خود برنامه می‌باید پیاده‌سازی شود و سیستم عامل قدرت لازم را ندارد.

جدول ۵-۳: معروف‌ترین توزیع کنندگان لینوکس

Name	Developer	Web Site
BlueCat Linux	LynuxWorks	http://www.lynuxworks.com
Hard Hat Linux	MontaVista Software, Inc	http://www.mvista.com
RT-Linux	FSMLabs	http://www.rtlinux.org
Coyote Linux	Prairie Wolf Software	http://www.coyotelinux.com
LinuxROM	KYZO	http://www.kyzo.com
FlightLinux	NASA	http://flightlinux.gsfc.nasa.gov
Midori Linux	Transmeta	http://midori.transmeta.com
White Dwarf Linux	EMJ Embedded Systems	http://www.emjembedded.com
Etlinux	PROSA	http://www.etlinux.org
Coollinux	Coollogic	http://www.coollogic.com
ELKS	Al Riddoch, et al	http://elks.sourceforge.net
Xteam Linux	Xteam Software Co.Ltd	http://www.xteamlinux.com.cn
PeeWeeLinux	Adi Linden	http://peeweelinux.com
JAILBAIT	http://jailbait.sourceforge.net	http://jailbait.sourceforge.net
EmbLin	Luc Hermans	http://www.dobit.com/emblin
ELinOS	SYSGO Real-Time Solutions	http://www.elinos.com
Royal Linux	ISDCorp	http://www.isdcorp.com
Siemens Industrial Linux	Siemens AG	http://siemens.de
Embedix	Lineo	http://www.lineo.com
Intrinsyc Linux	Intrinsyc Software	http://linux.intrinsyc.com
Roku OS	Roku LLC	http://www.rokulabs.com
EWRT	Portless Networks	http://www.portless.net/menu/ewrt
ucLinux	uClinux	http://www.uclinux.org
iPodLinux	iPodLinux Project	http://www.ipodlinux.org

۵–۵– توزیع های مختلف لینوکس

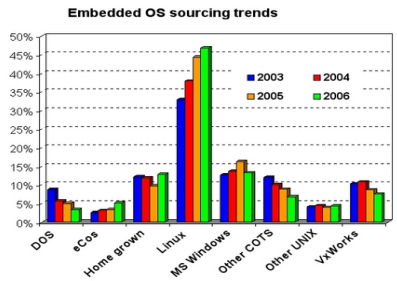
توزیع‌های عمومی مبنای کار بسیاری از پروژه‌ها قرار می‌گیرند، هرچند در حال حاضر توزیع کنندگان لینوکس‌های تعبیه شده هم وجود دارند.

در جدول(۵-۳) لیست بسیار کوچکی از معروف‌ترین توزیع کنندگان لینوکس تعبیه شده وجود دارد. لیست نسبتا کاملی از توزیع کنندگان در سایت http://www.linux.org/dist موجود است که چند صد توزیع کننده لینوکس با امکانات متنوع در آن ثبت شده‌اند.

۵–۶– آمارهای گوناگون

اگرچه یک آمار جامع و صحیح از نرخ استفاده از سیستم عامل‌های تعبیه شده، وجود ندارد ولی می‌توان ازآمارهای گوناگون دراین زمینه بعنوان نمونه استفاده کرد. گروه LinuxDevices.com در ششمین بررسی و آمارگیری سالیانه خود، نتایج قابل توجهی اعلام کرده است که مهم‌ترین عناوین آن در زیر بیان شده است. این آمارگیری به صورت چندین سوال از شرکت‌ها و گروه‌هایی که سیستم‌های تعبیه شده تولید می‌کرده‌اند، مطرح شده است و نتایج این سوالات با آمار سال‌های گذشته مقایسه شده‌اند.

یک سوال پرسیده شده از شرکت کنندگان این بود که کدام سیستم عامل در طول دو سال گذشته در شرکت شما در محصولات بلادرنگ استفاده شده است. نتایج سوال فوق به صورت آمار در شکل (۵-۶) آورده شده است.

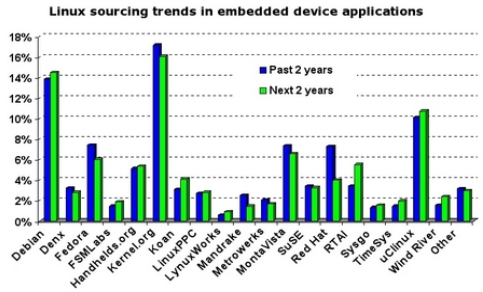


شکل۵- ۶: نمودار میزان استفاده از سیستم عامل های تعبیه شده

همان‌طور که مشاهده می‌شود، نسبت استفاده از لینوکس، در سیستم‌های تعبیه شده مختلف بسیار زیاد است و این نسبت در حال افزایش و رشد است. بیش از ۴۵٪ شرکت کنندگان از لینوکس در سال ۲۰۰۶ در سیستم‌های تعبیه شده خود استفاده می‌کرده‌اند. این

در حالیست که حدوداً ۱۰٪ افراد از سیستم عامل VxWorks که یکی از محبوب‌ترین سیستم عامل‌های متن بسته است، استفاده می‌کرده‌اند.

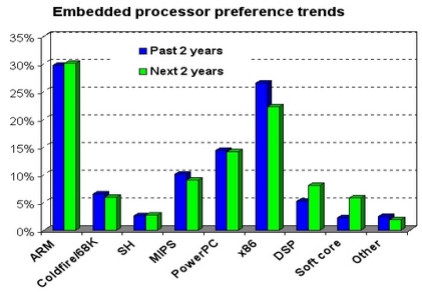
با توجه به رشد استفاده از سیستم عامل لینوکس در سیستم‌های تعبیه شده، چند نکته، حائز اهمیت هستند. اولین نکته این است که شرکت‌هایی که سیستم‌های تعبیه شده تولید می‌کنند، سیستم عامل تعبیه شده لینوکس خود را از کدام منبع یا توزیع کننده تهیه می‌کنند و یکی از سوالات پرسیده شده از شرکت کنندگان نیز همین بود که نتایج آمار سوال فوق در شکل زیر آورده شده است.



شکل۵-۷: نمودار میزان استفاده از کد منابع اولیه سیستم عامل Linux

در شکل(۵-۷) مشخص است که عمده کسانی که از سیستم عامل لینوکس در سیستم‌های تعبیه شده استفاده می‌کنند، مستقیماً از کد منبع موجود در سایت Kernel.org استفاده می‌کنند و توزیع کننده Debian و ucLinux به ترتیب در رتبه‌های دوم و سوم هستند.

نکته بسیار مهم دیگر، نوع سخت‌افزارها و پردازنده‌هایی است که این شرکت‌ها در سیستم‌های تعبیه شده خود استفاده می‌کنند، که این نکته هم به‌عنوان یک سوال از شرکت کنندگان سوال شد و نتایج سرشماری انواع سخت‌افزارها در شکل(۵- ۸) آورده شده است.



شکل ۵-۸ : نمودار مصرف پردازنده های مختلف در محصولات تعبیه شده

سری پردازنده‌های ARM و پس از آن سری پردازنده‌های x86 بیشترین کاربرد را در کاربردهای تعبیه شده در میان شرکت کنندگان داشته‌اند.

اگرچه آماری که LinuxDevices.com ارائه کرده است، چندان قابل استناد نیست ولی نشان دهنده رشد روز افزون لینوکس در سیستم‌های تعبیه شده است.

۵-۷- نتایج

بخشی از مهم‌ترین سیستم عامل‌های متن باز بلادرنگ و تعبیه شده موجود در دنیا در این قسمت بررسی شد که هر کدام ویژگی‌ها، نقاط قوت و ضعف خاص خود را دارند و می‌توان در هر پروژه با توجه به شرایط سخت‌افزار، شرایط نرم‌افزار و لیسانس محصول نهایی، یکی را انتخاب کرد.

در این میان نمی‌توان نتیجه‌گیری کلی به‌عنوان سیستم ایده‌آل برای کاربردهای بلادرنگ و تعبیه شده کرد، ولی با توجه به فعالیت‌های جهانی می‌توان گفت که امروزه لینوکس و مشتقات آن، در بسیاری از پروژه‌ها استفاده می‌شوند و لینوکس به یکی از داغ‌ترین مباحث در دنیا تبدیل شده است.

در کشور عزیزمان ایران، هم می‌توان با توجه به لیسانس GPL از لینوکس به سادگی استفاده کرد و آن را مبنای بسیاری از پروژه‌ها قرار داد. اگر در یک پروژه سخت‌افزار و یا محدودیت‌هایی وجود داشت که به ناچار لینوکس قابل استفاده نبود، می‌توان از سیستم‌های متن باز دیگری استفاده کرد. در بعضی پروژه‌ها که کد منبع می‌باید متن بسته قرار گیرد، می‌توان از سیستم عامل‌های خانواده BSD مانند NetBSD که به انواع سخت‌افزارها قابل انتقال می‌باشد، استفاده کرد.

فصل ششم – مشخصات سیستم عامل کارت هوشمند

درحال حاضر، بشر به این حقیقت دست پیدا کرده است که انتقال فیزیکی، زمان‌بر، پرهزینه و محدودکننده است و برای آنکه بتواند این مشکل را مرتفع کند، از ابزارهای مختلف سودجسته است. اما وسیله‌ای که بیش از وسایل دیگر مورداستفاده قرارگرفته و تاکنون بسیاری از مشکلات را حل کرده است و ازطرفی محدودیتهای کمتری نیز دارد ، انتقال اطلاعات ازطریق تکنولوژی‌های ارتباطی است. این انتقال، علاوه‌بر آنکه باعث پیشبرد فعالیت‌ها می‌گردد، محدودیت‌های انتقال فیزیکی را نداشته و حتی در مواردی بهتر ازآن عمل می‌کند. به عنوان مثال، در انتقال فیزیکی، امکان بروز اشتباه، دوباره کاری و... به وفور مشاهده می‌شود درحالی که در انتقال اطلاعات این موارد به حداقل مقدار خود می‌رسند. انتقال اطلاعات نیازمند یک تکنولوژی است که در جهان به عنوان فناوری اطلاعات شناخته می‌شود.

فناوری اطلاعات دارای این ویژگی است که مرز علم و دانش را برداشته و از تجمع دانایی در یک فرد یا یک مجموعه جلوگیری می‌کند. با این تکنولوژی، ارتباط کشورها و مردمان آن بیشتر و نزدیکتر شده و جهان به سمت یک دهکده جهانی پیش می‌رود و درنهایت باعث می‌شود که مکان، معنی فعلی خود را ازدست داده و محدودیت‌های مکانی ازبین برود.

از آنجایی که این تکنولوژی باعث می شود که تمامی حرفه‌ها به صورت جهانی بتوانند از آن بهره‌گیری کنند، لزوم آشنایی و استفاده از آن ثابت می‌شود. به‌عنوان مثال، تولیدکنندگان برای تمامی مراحل خرید مواد اولیه، تولید ریز قطعات، مونتاژ قطعات و فروش محصولات خود نیاز به بهره‌گیری مناسب از فناوری اطلاعات دارند چرا که چنانچه این بهره‌گیری به صورت مناسب وجود نداشته باشد، باعث از دست رفتن فرصت‌های مختلف می‌گردد.

روند فناوری اطلاعاتی این حقیقت را نشان می‌دهد که یا باید با این تکنولوژی آشنا شده و حداکثر استفاده را از آن کرد؛ یا اینکه چشم و گوش را به روی آن بست و آن را نادیده گرفت. نگاهی گذرا به چگونگی پیشرفت کشورهای توسعه یافته و روند پیشرفت آنها نشان می‌دهد که این کشورها در این تکنولوژی پیشقدم بوده و از آن به نحو مناسب استفاده می‌کنند. درحالی که سایر کشورها به نحو مناسب از این تکنولوژی استفاده نمی‌کنند.

با استفاده از این تکنولوژی، پیاده‌سازی سیستم‌های مختلف که قبلاً بدون دستیابی به چنین ابزاری، مشکل به نظر می‌رسید؛ راحت‌تر و عملی‌تر شده است . به عنوان مثال، شرکت‌های مختلف برای انجام مقایسه نیاز به شناسایی رقبا و نقاط قوت وضعف آنها دارند که

به نظر می‌رسد با جهانی شدن این تکنولوژی، این شناسایی بسیار ساده‌تر شده و به وسیله انتقال اطلاعات این شرکت‌ها می‌توانند به نحو مناسب عملیات مقایسه را انجام داده و موقعیت خود را ارتقا بخشند. این موضوع بویژه درایران حائز اهمیت است، چرا که ایران در وضعیتی است که اگر از فواید این تکنولوژی بهره نبرد قطعاً از کشورهای مختلف عقب خواهد افتاد.

بایستی توجه داشت که برای دستیابی به این اهداف، استفاده از فناوری اطلاعات لازم است ولی کافی نیست. کارت هوشمند یکی از جدیدترین ابزارهای راه یافته به دنیای فناوری اطلاعات است. این کارت‌ها شامل یک ریزپردازنده یا قطعه حافظه است. این قطعه داده‌های الکترونیکی و برنامه‌هایی که از ویژگی‌های امنیتی پیشرفته برخوردارند را در خود ذخیره کرده است و زمانیکه کارت در کنار یک خواننده اطلاعات قرار می‌گیرد، قادر به اجرای برنامه‌های مختلفی است. کارت‌های هوشمند قابلیت حمل داده، امنیت و راحتی را با خود به ارمغان آورده‌اند. کارت‌های هوشمند امروزه در تلفن، حمل و نقل، بانکداری و ... کاربرد دارند و بزودی در برنامه‌های اینترنتی نیز استفاده خواهند شد. این کارت‌ها امروزه در ژاپن و اروپا به طور گسترده استفاده می‌شوند و در آمریکا نیز مورد استقبال مردم قرار گرفته‌اند. در حقیقت پیشرفت و گسترش صنعت کارت هوشمند بسیار سریع می‌باشد.

۶-۱- کارت هوشمند چیست ؟

یک کارت هوشمند کارتی پلاستیکی در اندازه کارت‌های اعتباری می‌باشد که یک قطعه الکترونیکی کوچک را در خود جای داده است. این قطعه کوچک می‌تواند یک حافظه کوچک و یا یک ریزپردازنده که شامل حافظه و CPU است، باشد.

کارت‌های حافظه معمولاً برای کارت‌های تلفن استفاده می‌شود در حالیکه کارت‌های ریزپردازنده می‌توانند برای برنامه‌های مختلف استفاده گردند. گرچه هردوی این کارت‌ها داده ذخیره شده را در خود دارند، کارت ریزپردازنده می‌تواند این داده‌ها را پردازش کند زیرا یک CPU، RAM و یک سیستم عامل واقع در حافظه فقط خواندنی (ROM) را نیز در خود دارد.

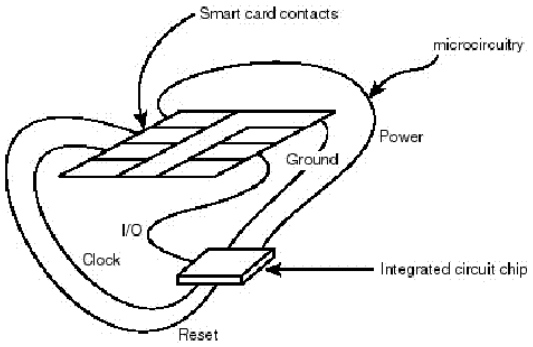
هر مجموعه از مدارهای داخل کارت هوشمند دارای ۸ گذرگاه ارتباطی فلزی بوده که هر یک دارای استاندارد بین المللی می‌باشند:

- VCC (Power supply voltage) گذرگاه برای تامین ولتاژ و برق RST(Reset Microprocessor of Smartcard) برای راه اندازی مجدد پردازنده کارت هوشمند.
- CLK(Clock signal) سیگنال ساعت
- GND اتصال زمین
- VPP(Programming or write voltage) گذرگاه دستورات ذخیره اطلاعات
- I/O(Serial input and output) ورودی/خروجی
- (RFU) دو گذرگاه جهت استفاده‌های آینده

هنگامی که یک کارت در داخل دستگاه کارتخوان (Card Acceptance Device – CAD) قرار میگیرد ، پینهای موجود در داخل کارتخوان روی گذرگاههای کارت قرار گرفته و مجموعه، آماده تبادل اطلاعات میباشد. هر بار که کارت درون کارتخوان قرار بگیرد ، کارت مجدداً راه‌اندازی شده و آماده دریافت فرامین جدید می‌باشد.

انواع مختلف کارت‌های هوشمند که امروزه استفاده می‌شود، کارت‌های تماسی ، بدون تماسی و کارت‌های ترکیبی هستند.

کارت‌های تماسی دارای یک صفحه طلایی هستند که باید با دستگاه خواننده(Reader) تماس پیدا کرده و اطلاعات آن مورد بازخوانی قرار گیرد. کارت‌های بدون تماس ، یک آنتن سیم پیچی درون خود دارا هستند که همانند چیپ درداخل کارت ، گنجانده شده است . این آنتن درونی اجازه انجام ارتباطات و ردوبدل کردن اطلاعات را فراهم می‌آورد. برای چنین ارتباطی، بایستی علاوه بر اینکه زمان ارتباط کاهش یابد، راحتی نیز افزایش پیدا کند. مزیتی که این کارت نسبت به حالت قبل دارد این است که نیاز به کارت خوان ندارد اما باید توجه داشت که در این مورد بایستی ارتباط اولیه توسط آنتن حتما برقرار گردد، درغیر این صورت نمی‌توان از کارت استفاده کرد. کارت‌های غیر تماسی گرانتر از تماسی بوده و امروزه بیشتر در حمل و نقل و یا ورودی ساختمان‌ها مورد استفاده قرارمی‌گیرند.



شکل۶-۱: کارت‌های هوشمند تماسی

کارت‌های ترکیبی، به عنوان هم کارت‌های تماسی و هم کارت‌های بدون تماس عمل می‌کنند و در حقیقت داخل این نوع کارت‌ها هم چیپ الکترونیکی و هم آنتن وجود دارد وچنانچه کارت‌خوان وجود داشته باشد از کارت خوان می‌توان استفاده کرد و چنانچه وجود نداشته باشد، از آنتن کارت می‌توان ارتباط را برقرار کرد.

۶-۱-۱-کاربردهای کارت هوشمند

با داشتن کارت‌های هوشمند دیگر نیازی به داشتن کارت‌های دیگر نیست. کاربردهای کارت‌های هوشمند به شرح زیر هستند.

۶-۱-۱-۱- مراقبت‌های پزشکی

با کارت‌های هوشمند، اطلاعات پزشکی و سایر اطلاعات مربوط به بیماری‌های اشخاص همچنین اطلاعاتی راجع به سوابق پزشکی و دارویی افراد در قسمت مشخصی از کارت با سطح دسترسی معین، همچنین اطلاعات مربوط به بیمه افراد در کارت ذخیره می‌گردد. ترکیب اطلاعات پزشکی و اطلاعات بیمه در یک کارت، باعث حذف کاغذبازی گردیده و مراقبت پزشکی بیشتر و موثر را روی شخص فراهم می‌آورد. کارت هوشمند، این امکان را فراهم می‌آورد که اشتباهات مختلف که در انتقال اطلاعات توسط کاغذ پیش می‌آید به حداقل ممکن برسد. به عنوان مثال، در یک نسخه ممکن است داروخانه درخواندن دستخط اشتباه کند یا اینکه مفاهیم به بیمار درست منتقل نشود درصورتی که با کارت هوشمند، اطلاعات لازم در آن ثبت گردیده و قسمت‌های مختلف مانند داروخانه‌ها، آزمایشگاه‌ها و سایر قسمت‌های لازم اطلاعات

مربوطه را به صورت کامل و صحیح از روی کارت دریافت می‌کنند. در بقیه موارد نیز ازقبیل سوابق پزشکی، از روی کارت هوشمند می‌توان به راحتی به این اطلاعات دست پیدا کرد. به عنوان مثال، دندانپزشک با مراجعه به سوابق پزشکی درج شده در کارت هوشمند می‌تواند دندان‌های بیمارش را به راحتی درمان کند. همچنین در اورژانس نیز، فقط یک کارت هوشمند، اطلاعات لازم را به پزشک معالج می‌رساند. سود دیگری که این کارت دارد، این است که عکس‌های اشعه X که ازشخص گرفته می‌شود می‌تواند در داخل کارت هوشمند ثبت گردد. بنابراین انتقال این عکس‌ها به راحتی میسر می‌گردد. این مورد، زمان و هزینه را برای بیماران و پزشکان کاهش می‌دهد. لازم به ذکر است که در استرالیا و آلمان بیش از ۸۰ میلیون کارت به همین منظور برای افراد صادر شده است.

۶-۱-۱-۲- اطلاعات دانشگاهی

اغلب دانشگاه‌ها هم اکنون یک کارت هوشمند دانشگاهی جهانی را برای تبادل تمامی اطلاعات دانشگاهی به کار می‌برند. سوابق تحصیلی، اطلاعات مالی(صورتحسابی)، اطلاعات کتابخانه، ژتون‌های غذا و غیره همگی در یک کارت هوشمند قرار می‌گیرد که می‌تواند به صورت موثر، اطلاعات مذکور را دربرداشته باشد.

این کارت، یک محیط بدون پرداختی را در محیط دانشگاه فراهم می‌آورد که دانشجویان به علت راحتی از آن استفاده می‌کنند. به عنوان مثال، وقتی یک نفر فارغ التحصیل دانشگاهی برای کار به مجموعه‌ای مراجعه می‌کند، برای آنکه آن مجموعه از سوابق تحصیلی فرد اطلاع حاصل کند، با دریافت رمز عبور از فارغ التحصیل و گرفتن اطلاعات کارت هوشمند فرد می‌توانند تمامی اطلاعات لازم ازقبیل محل تحصیل، مدرک تحصیلی، معدل، ریز نمرات، استادان و... را دریافت کند و براساس آن تصمیم بگیرد.

۶-۱-۱-۳- فعالیت‌های اداری – سازمانی

کارت‌های هوشمند این امکان را برای افراد فراهم می‌کند که افراد مختلف یا با مراجعه حضوری و یا با استفاده از کارت خوان یا با استفاده از کارت‌های بدون تماس و با استفاده ازسایت مربوط به اداره/سازمان موردنظر، اطلاعات و درخواست خود را به آن سازمان ارائه کنند تا فعالیت درخواستی فرد باتوجه به اطلاعات مندرج در کارت هوشمند انجام پذیرد. به عنوان مثال، یک فرد که تقاضای صدور گذرنامه می‌کند می تواند با استفاده ازکارت هوشمند اطلاعات

لازم را دراختیار سازمان صادرکننده آن قرار دهد و آنها با بررسی اطلاعات مندرج در کارت اقدام به صدور گذرنامه روی همان کارت کنند و فرد به راحتی می‌تواند این تسهیلات را دریافت کند. دریافت بلیت نمونه دیگری از کاربرد کارت هوشمند است.

۶-۱-۱-۴- اطلاعات شخصی (شناسایی)

با استفاده از کارت‌های هوشمند، اطلاعات شناسایی افراد در داخل آنها ثبت می‌گردد و در صورت نیاز دراختیار افراد/سازمان‌های مربوطه قرار داده می‌شود. این اطلاعات می‌تواند شناسنامه، کارت‌های شناسایی، گذرنامه و... را شامل شود.

۶-۱-۱-۵- کارت تلفن

همان‌گونه که در کشور ما نیز این کارت‌ها مورد استفاده قرار می‌گیرند، می‌توانند درکارت‌های هوشمند گنجانده شده و سایر تسهیلات جانبی آن نیز از قبیل اضافه کردن اعتبار و... را دریافت کرد. درحال حاضر بیش از ۱۰۰ کشور جهان ، سیستم پرداخت تلفن عمومی را از حالت سکه‌ای حذف کرده و یا کاهش داده‌اند و به جای آن از کارت تلفن استفاده می‌کنند.

۶-۱-۱-۶- کارت GSM

GSM کارت‌هایی هستند که در موبایل‌ها استفاده می‌شوند و خود نوعی کارت هوشمندند که می‌تواند مرتبط با کارت هوشمند هر فرد باشد.

۶-۱-۱-۷-کارت بانکی (اعتباری)

اطلاعات دیگری که می‌تواند در کارت‌های هوشمند وجود داشته باشد، اطلاعات بانکی است که هر فرد می‌تواند اطلاعات مربوط به بانک‌های مختلف را به همراه اطلاعات اعتباری فروشگاه‌های مختلف در آن نگهداری کرده و مورد استفاده قرار دهد. به عنوان مثال، شماره حساب فرد، میزان موجودی و همچنین سایر اطلاعات می‌تواند در کارت هوشمند ذخیره شده و مورداستفاده قرار گیرد. لازم به ذکر است که در آلمان درحدود ۴۰ میلیون کارت بانکی با این مدل انتشار یافته است. در سنگاپور و پرتغال نیز این نوع فعالیت‌ها چشمگیر است.

۶-۱-۱-۸- کارت‌های تلویزیون و بازی

درحال حاضر برخی از کانال‌های تلویزیون برای دریافت تصاویر، نیاز به کارت‌های اعتباری دارد. یکی دیگر از کاربردهای کارت‌های هوشمند همین مورد است و هر فرد بسته به نیاز می‌تواند این امکان را در کارت هوشمند خود ذخیره کرده و مورداستفاده قرار دهد. این مورد برای برخی از بازی‌های کامپیوتری (یا برخی از سایت‌های اینترنتی) کاربرد دارد. مثلاً در آمریکا ۴۰۰ هزار کارت به این منظور تهیه شده است .

۶-۱-۲- ویژگی‌های کارت‌های هوشمند

۶-۱-۲-۱- صداقت

با توجه به برنامه کاربردی که در کارت هوشمند وجود دارد، برای کاربردهای مختلف، به صورت برنامه‌ریزی شده عمل می‌گردد. به عنوان مثال، برای برخوردار بودن از تسهیلات فروشگاههای مختلف مانند تخفیف در اثر خرید بیش از مقدار معین، کارت‌های هوشمند اطلاعات لازم را ذخیره کرده و در زمان لازم، تسهیلات پیش‌بینی شده را درنظر می‌گیرد و این امر سبب می‌گردد که بتوان بیشتر به ارتباطات خود اعتماد کرد. همچنین ویژگی حاضر در کارت‌های هوشمند، امکان ایجاد کارت‌هایی مخصوص بچه‌ها را فراهم می‌آورد که این کارت‌ها از کارت پدر تغذیه می شوند و محدودیت‌های اعمال شده روی کارت توسط پدر، درهنگام خرید فرزند فراهم می شود و این امر، از خرید برخی وسایل مانند دارو توسط فرزندان جلوگیری می‌کند.

۶-۱-۲-۲- کنترل دسترسی

امکان دسترسی به این کارت‌ها، را می‌توان با توجه به نیاز تعریف کرد و هر شرکت یا سازمان یا هر شخص دیگری فقط امکان دسترسی به اطلاعات مورد نیاز خود را داراست و این سطح دسترسی قابل تعریف است .

از آنجایی که کارت‌های هوشمند دارای (File Allocation Table – FAT) هستند می‌توان شاخه‌ها، کاربران وگروه کاربران را درآن تعریف کرد و دسترسی به اطلاعات این پرونده‌ها و شاخه‌ها از طریق (Access Control List – ACL) مدیریت می‌شوند. برای مثال، سازمانی که فقط نیاز به اطلاعات مشخصی دارد، دارنده کارت، شماره عبور آن قسمت را به تنهایی از ACL

استخراج کرده و در اختیار آن سازمان قرار می‌دهد و بدین ترتیب اطلاعات کارت در اختیار آن سازمان قرار می‌گیرد و چون سازمان مورد نظر، فقط از شماره عبور قسمت مورد درخواست آگاهی دارد، فقط به اطلاعات لازم دسترسی خواهد داشت.

۶-۱-۲-۳- امنیت اطلاعات

- احراز هویت (Authentication)؛ این سندیت ازطریق PIN یا DES فراهم می‌شود، یکی از روش‌ها می‌تواند اثرانگشت باشد.
- .اجازه ورود (Authorization)؛ نیازمندی‌های هر کاربر را باتوجه بـه ACL بررسـی کرده و باتوجه به آن ، مجوز استفاده از اطلاعات صادر می‌گردد.
- رمز نویسی(cryptography)؛می توان روی کارت هوشمند از طریق الگـوریتم‌هـای RSA,SHA,DES,TRIPLE DESرمز نویسی کرد.

۶-۱-۲-۴- میزان حافظه

کارت‌های هوشمند در حال حاظر دارای حافظه ۶۱ و۲۳ کیلو بایت هستند و به زودی این رقم به ۸۴ و۴۶ کیلوبایت افزایش خواهد یافت.

۶-۱-۲-۵- امکان برنامه نویسی

کارت‌های هوشمند به علت برخوردار بودن از APIامکان برنامه نویسی و استفاده از برنامه‌های کاربردی در داخل کارت را فراهم میاورد.این کارت‌ها به افراد اجازه می‌دهد دستوراتی مانند ایجاد فایل، خواندن، نوشتن و تصدیق فایل را روی برنامه کاربردی قرار دهد. این برنامه‌های کاربردی می توانند به هر زبانی تهیه گردد.

لازم به ذکر است این برنامه‌های کاربردی با توجه به نیاز هر فرد/سازمان می تواند تعریف شده و پس از مدت مورد نیاز، حذف گردد و این مورد فقط برای زمانی است که حالت خاصی برای فرد خاصی پیش آید و برای ثبت اطلاعات از این مورد استفاده شود.

۶-۱-۲-۶- پیروی از استاندارد

برای آنکه کارت‌های هوشمند مربوط به کشورهای مختلف از شکل و شیوه یکسانی برخوردار باشند (تا کارت‌ها در تمام نقاط جهان قابل استفاده باشند) کارت‌های هوشمند ازاستاندارد جهانیISO 7816 استفاده می‌کنند که این استاندارد شامل موارد زیر می‌گردد:

الف – ویژگی‌های فیزیکی

ب - ابعاد و محل اتصالات

ج – پروتکل‌های انتقال و علائم الکترونیکی

د - دستورات مربوطه

ه - سیستم شمارش و روش ثبت

و - عناصراطلاعاتی

درهر حال، کارت‌های هوشمند بایستی برآورده کننده الزامات این استاندارد باشند.

۶-۱-۳-تفاوت کاربرد فناوری اطلاعات با روش سنتی

درحال حاضر بیشتر اطلاعات به جای انتقال فیزیکی از طریق فناوری اطلاعات منتقل می‌گردند. این امر باعث کاهش زمان، هزینه و نگهداری امنیت اطلاعات می‌شود. هم اکنون اکثر کشورها و از آن جمله ایران نیاز به بهره‌گیری از فناوری اطلاعات را کاملاً حس کرده و درجهت تامین این ابزارها قدم برمی‌دارند. تفاوت فناوری اطلاعات و روش سنتی در این است که در روش سنتی، در اغلب موار اطلاعات به طور فیزیکی منتقل می‌شوند. مثلاً در تجارت، ابتدا بازاریابی با حضور بازاریاب و سپس فرستادن نمونه محصول و درنهایت فرستادن محصول درخواست شده انجام می‌شود؛ درحالی که با تجارت الکترونیکی بیشتر اطلاعات رد و بدل شده و فرآیند فروش سریعتر و کم هزینه‌تر انجام می‌شود. با این روش می‌توان خریداران و فروشندگان بهتری را در زمان کم شناسایی کرد.

اما در روش سنتی به جای حمل کارت‌ها، دفترچه‌ها و اطلاعات کاغذی، کافی است از یک کارت هوشمند که تمامی اطلاعات لازم مربوط به فرد را داراست، استفاده کرد.

با استفاده از کارت هوشمند مشکل گم شدن برخی مستندات، فراموشی و اشتباه به حداقل می‌رسد. در ثانی پردازش برخی از اطلاعات به راحتی میسر است چرا که این کارت‌ها به

صورت استاندارد تهیه شده و لذا اطلاعات دریافت شده توسط افراد/موسسات دیگر (برخلاف روش سنتی)، نیاز به تفسیر ندارد.

۶-۱-۴- زمینه‌های جهانی گسترش کاربرد فناوری اطلاعات در جهان آینده

از آنجایی که درحال حاضر محدودیت‌های بین کشورها مانند واحد پولی، مرزهای کشوری و غیره هنوز به طور کامل حل نشده است و از طرفی برای اینکه کارت‌های هوشمند هرچه بهتر و راحت‌تر مورد استفاده قرار گیرند؛ نیاز به حل این مشکلات محسوس است. چنانچه این مشکلات حل شود، کاربرد کارت‌های هوشمند بهتر و مفیدتر خواهد بود. مثلاً افراد در مسافرت‌های خارجی، نیاز به حمل پول نداشته و فقط با استفاده از اعتباری که در کشور مبدا در کارت هوشمند قرار داده‌اند می‌توانند نیازهای مالی خود را در آن کشور برطرف کنند.

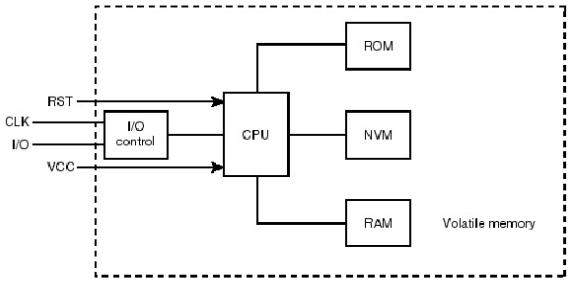
۶-۱-۵- موانع پیش رو برای پیاده سازی کارت‌های هوشمند در ایران

درحال حاضر انواع مختلفی از کارت‌های هوشمند در ایران مورداستفاده قرار می‌گیرند، اما این ایراد بر آنها وارد است که از نظر نوع، شکل و کاربرد دارای استاندارد یکسانی نیستند و هریک از روش‌ها و استانداردهای مربوط به خود استفاده و پیروی کرده است. در واقع نیاز به رعایت استانداردهای جهانی در این مورد احساس می‌شود.

مشکلات دیگری که در این راستا وجود دارد، یکی مساله ارتباط واحد پول کشور ما با سایر کشورها است که بایستی به نوعی حل شود و مساله دیگر، نیاز داشتن به کارت‌خوان‌های یکسان در کل کشور است، که خود این کارت‌خوان نیز باید براساس استاندارد ISO 7816 باشد. مشکل اساسی دیگر این است که در کشور باید فرهنگ استفاده از کارت هوشمند جا بیفتد. بدین معنی که باید به همه افراد جامعه، آموزش داد که برای از بین بردن صف‌های مختلف در موسسات، از این کارت استفاده کنند. یعنی با این کار بایستی نحوه گردش کارها در موسسات و نقاط اشتراک آن با افراد کاملاً مشخص و تعیین گردد.

۶-۲- سخت افزار کارت‌های هوشمند

کامپیوتر سوار بر یک کارت هوشمند یک مدار مجتمع ساده و کوچک است که شامل واحد پردازشگر مرکزی (CPU)، سیستم حافظه و خطوط ورودی/خروجی است.



شکل۶-۲: اجزاء کامپیوتر کارت هوشمند

۶-۲-۱- سیستم حافظه

کارت‌های هوشمند دارای معماری حافظه‌ای هستند که برای اغلب برنامه نویسان ناآشنا است. در واقع یک کارت هوشمند دارای سه نوع حافظه است: حافظه فقط خواندنی (ROM)؛ حافظه غیرفرار (nonvolatile memory-NVM) و حافظه دسترسی تصادفی (RAM).

حافظه فقط خواندنی (ROM)، محل نگهداری سیستم عامل کارت هوشمند می‌باشد. برنامه و اطلاعات زمانی که کارت ساخته می‌شود در حافظه فقط خواندنی، از طریق برنامه‌های رمزنگاری و ریاضی خاصی ذخیره می‌شوند و قابل تغییر نمی‌باشند.

NVM جایی است که داده‌های متغیر مانند شماره حساب، موجودی حساب و... ذخیره می‌شوند. داده‌ها می‌توانند توسط برنامه‌های کاربردی از NVM خوانده شده و یا بر روی آن نوشته شوند. اما NVM نمی‌تواند همانند RAM مورد استفاده قرار گیرد. اگرچه می‌توان بر روی NVM نوشت ولی ماهیت این عمل با ماهیت نوشتن بر روی RAM متفاوت است. NVM نام خود را از این واقعیت می‌گیرد که با قطع برق محتوای کارت از بین نمی‌رود.

بر روی یک کارت هوشمند RAM وجود دارد ولی حجم آن زیاد نیست. این نوع از حافظه از دیدگاه برنامه نویسان ارزشمندترین منبع موجود در یک کارت است. حتی زمانیکه از زبان‌های سطح بالای برنامه‌نویسی استفاده می‌شود؛ برنامه نویس از اهمیت نگهداری متغیرهای موقت آگاه است. این حافظه RAM موجود در کارت فقط به وسیله برنامه‌های کاربردی برنامه نویس استفاده نمی‌شود؛ بلکه از آن در سایر برنامه‌های کارت نیز استفاده می‌گردد. لذا برنامه‌نویس نه تنها باید به میزان RAMی که برنامه‌اش اشغال می‌کند توجه نماید؛ بلکه باید به RAMی که برنامه‌های فراخوانی شده توسط برنامه‌اش نیز استفاده می‌کنند توجه کند.

۶-۲-۲- واحد پردازشگر مرکزی

در یک کارت هوشمند ۸ بیتی واحد پردازشگر مرکزی Motorola 6805 و Intel 8051 است. CPUها با سرعت ۴۰۰۰۰۰ دستور در ثانیه کار می کنند و امروزه در نسل جدید کارت‌ها به ۱۰۰۰۰۰۰ دستور در ثانیه نیز رسیده است. تقاضا برای رمزنگاری قویتر در کارت‌های هوشمند موجب شده است نرم‌افزار این کارت‌ها نتیجه را در زمان کمتری حاضر کند. ۱ تا ۳ ثانیه زمانی است که یک کارت هوشمند برای کل علیات خود باید صرف نماید؛ در صورتیکه یک الگوریتم رمزنگاری RSA با کلید ۱۰۲۴ بیتی به ۱۰ تا ۲۰ ثانیه زمان برای پردازش بر روی یک پردازشگر کارت نیاز دارد. در نتیجه برای رسیدن به زمان مورد نظر و قدرت رمزنگاری بیشتر؛ از چند پردازشگر همزمان در کارت‌ها استفاده می گردد.

طی ۲۰ سال پیشرفت سرعت کارت‌های هوشمند افزایش یافته است زیرا حجم حافظه افزایش یافته و معماری پردازشگر از ۸ بیت به ۱۶ یا ۳۲ بیت ارتقا یافته است.

۶-۲-۳- ورودی/خروجی

کانال ورودی/خروجی در یک کارت هوشمند یک کانال یک طرفه سریال است. سخت‌افزار کارت قادر به handle کردن داده‌ها با سرعت حداکثر ۱۱۵۲۰۰ بیت در ثانیه است؛ ولی خواننده‌های کارت هوشمند با سرعت بسیار کمتری ارتباط برقرار می‌کنند.

پروتکل ارتباطی بین میزبان و کارت هوشمند بر پایه رئیس(master) و برده(slave) است و رئیس میزبان بوده و برده کارت می‌باشد. میزبان دستورات را به کارت فرستاده و منتظر پاسخ می‌ماند. کارت اطلاعاتی برای میزبان نمی‌فرستد مگر در حالیکه میزبان در انتظار پاسخ به یک دستور است.

۶-۳- نرم‌افزار کارت هوشمند

نرم‌افزارهای کارت‌های هوشمند اساساً بر دو نوع است، نوع اول نرم‌افزار میزبان است که روی کامپیوتری که به کارت متصل می‌شود اجرا می‌گردد. نام دیگر این نوع نرم‌افزار، سمت خواننده است. نوع دوم نرم‌افزار کارت است که روی خود کارت اجرا می‌گردد و نرم‌افزار سمت کارت نامیده می‌شود.

نرم‌افزار اصلی کارت‌های هوشمند مربوط به میزبان است که برای کامپیوترهای شخصی و سرورهای workstation که به کارت متصل می‌شوند نوشته می‌شود و کارت را به سیستم‌های بزرگتر وصل می‌نمایند. این نرم‌افزارها شامل برنامه‌های کاربران نهایی، برنامه‌های سیستمی که خواننده‌ها را با میزبان مرتبط می‌سازند، و برنامه‌های کاربردی که برای مدیریت کارت است، می‌باشند.

نرم‌افزارهای میزبان عموماً با زبان‌های سطح بالای برنامه نویسی مانند C، C++، Java، BASIC، COBOL، Pascal و FORTRAN نوشته می‌شوند و به وسیله کتابخانه‌های موجود و درایورهای خاص به کارت هوشمند و خواننده آن متصل می‌گردند.

در مقابل نرم‌افزار کارت هوشمند معمولاً با زبان‌های قدرتمند محاسباتی مانند Java، زبان‌های ماشین مثل Forth و یا اسمبلی نوشته می‌شوند. نرم‌افزار کارت هوشمند معمولاً شامل سیستم عامل، برنامه‌های کاربردی و برنامه‌های مورد نیاز می‌شود.

برنامه‌های سیستمی کارت به زبان‌های سطح پایین ماشین نوشته می‌شوند و برای انجام عملیات پایه‌ای در کارت هوشمند استفاده می‌شوند.

برنامه‌های میزبان و برنامه‌های کارت اساساً از لحاظ پایه‌ای با یکدیگر متفاوت هستند. برنامه‌های کارت بر روی محتویات کارت و نحوه دستیابی به آن و ایمن نگه داشتن این محتویات تمرکز کرده است. در حالیکه برنامه‌های میزبان با کارت‌های مختلف در ارتباط هستند و با انواع این کارت‌ها ارتباط برقرار می‌کنند.

نرم‌افزار کارت داده‌ها را پردازش کرده و سیاست‌های ایمن‌سازی را عملیاتی می‌کند. برای مثال یک برنامه که در حال اجرا شدن بر روی کارت است، به شماره حساب بانکی ذخیره شده بر روی کارت دسترسی ندارد تا زمانیکه شماره رمز درست را وارد کرده باشد. نرم‌افزار کارت هوشمند دسترسی قانونی و ایمن به داده‌های ذخیره شده روی کارت را تامین می‌کند؛ و تنها از محتوای کارت و ابزارهایی که قصد دستیابی به این محتوا را دارند آگاهی دارد. نرم‌افزار میزبان، کارت‌های هوشمند و کاربران آن را به سیستم‌های بزرگتر متصل می‌کند. برای مثال نرم‌افزاری که بر روی یک دستگاه ATM در حال اجراست از اطلاعات موجود در کارت وارد شده استفاده می‌کند و مشتری بانک را شناسایی کرده و آن را به حساب مورد نظر خود وصل می‌کند.

۶–۴– استانداردهای کارت‌های هوشمند

استاندارد کارت‌های تماسی سری ISO 7816 است و برای کارت‌های غیرتماسی ISO 14443 است. این استانداردها ویژگی‌های فیزیکی، الکتریکی، مکانیکی و برنامه نویسی کارت‌ها را مشخص می سازد. در زیر به لیستی از استانداردهای کارت‌های تماسی اشاره شده است.

۱) IS 7816 – 1 : ویژگی‌های فیزیکی (۱۹۸۷)
۲) IS 7816 – 2 : ابعاد و مکان تماس (۱۹۸۸)
۳) IS 7816 – 3 : سیگنالهای الکترونیکی و پروتکل ارسال (۱۹۸۹)
۴) IS 7816 – 4 : دستورات و پاسخها - Commands and Responses (۱۹۹۵)
۵) IS 7816 – 5 : ثبت برنامه‌های تعیین هویت (۱۹۹۴)
۶) IS 7816 – 6 : عناصر داده‌ای برای مبادله اطلاعات (۱۹۹۵)
۷) IS 7816 – 7 : زبان Query کارت‌های هوشمند (۱۹۹۸)
۸) DIS 7816 – 8 : دستورات امنیتی
۹) DIS 7816 – 9 : دستورات اضافه شده
۱۰) DIS 7816 – 10 : کارت‌های همزمان

۶–۵– سیستم عامل های کارت های هوشمند

هر کارت هوشمندی دارای سیستم عامل است. این سیستم عامل خصوصیات نرم-افزاری خاص است که در حافظه ROM فقط خواندنی نگهداری شده و فراهم آورنده مجموعه امکاناتی از قبیل دسترسی امن به اطلاعات کارت، احراز هویت و رمزنگاری می‌باشد. تعداد کمی از کارت‌های هوشمند مانند کامپیوترهای شخصی امکان ذخیره اطلاعات برنامه‌های اجرا شده را در داخل کارت دارند. این امر سبک عظیمی در توسعه سیستم عامل‌های کارت‌های هوشمند بوجود آورده است.

سیستم عامل تراشه‌های کارت‌های هوشمند(که به اختصار COS یا Mask نامیده می‌شود)، مجموعه‌ای از دستورالعمل‌ها است که بصورت ماندگار در حافظه ROM قرار دارد. مانند سیستم عامل‌های DOS و WINDOWS، سیستم عامل‌های کارت‌های هوشمند نیز به برنامه‌های خاص دیگری وابسته نیستند و در بالاترین سطح نرم‌افزاری در کارت هوشمند قرار داشته و توسط دیگر برنامه‌ها استفاده میگردند.

سیستم عامل‌های تراشه‌های کارت‌های هوشمند به دو خانواده تقسیم می‌شوند :

- سیستم عامل های همه منظوره جهت کارت‌های هوشمند که دارای قابلیت دستورالعمل‌های عمومی بوده و در ابعاد گوناگون نیازهای برنامه‌های مختلف را پوشش میدهند.
- سیستم عامل‌های تک منظوره که دستورالعمل‌های آنها برای برنامه‌های خاص استفاده شده و حتی می توانند شامل خود برنامه‌های کاربردی نیز باشند. از این نمونه می‌توان به کارت‌هایی که برای پرداخت‌های الکترونیکی مالی استفاده می‌شود، اشاره کرد.

عملیات اصلی که COS به طور معمول برای انواع کارت هوشمند شامل می‌شود به شرح زیر است:

- مدیریت مبادلات داده‌ها بین کارت و دنیای خارج بر پایه پروتکل تبادل اطلاعات
- مدیریت فایل‌ها و داده‌های ذخیره شده در حافظه
- کنترل دسترسی به اطلاعات و عملیات(خواندن، نوشتن، و به‌روزرسانی اطلاعات)
- مدیریت امنیت کارت و الگوریتم‌های رمزنگاری
- مدیریت وقفه‌های احتمالی، ثبات داده‌ها و بر طرف کردن errorهای پیش آمده
- مدیریت فازهای مختلف در چرخه حیات کارت (ساخت کارت، شخصی شدن کارت، دوران فعال بودن کارت و غیرفعال شدن کارت)
- ارتباط پایه بین ترمینال کارت و خود کارت از قاعده master/slave پیروی می‌کند. ترمینال یک دستور برای کارت ارسال می‌کند، کارت دستور را اجرا می‌نماید، و در صورت موجود بودن جواب آن را برای ترمینال ارسال کرده و در انتظار دستورات بعدی ترمینال می‌ماند.

علاوه بر توصیف ویژگی‌های فیزیکی کارت، استانداردهای کارت‌های هوشمند مانند ISO 7816 و CEN 726 طیف گسترده‌ای از دستوراتی را که کارت‌های هوشمند می‌توانند اجرا کنند توصیف می‌کنند. اکثر سازندگان کارت‌های هوشمند، کارت‌هایی با سیستم عامل‌هایی که توانایی اجرای اغلب این دستورات را دارند عرضه می‌نمایند.

در ابتدای پیدایش COS، سازندگان به ازای هر برنامه مورد نظر مشتری می بایست یک سیستم عامل خاص برای آن تدارک می‌دیدند. در نتیجه کارت‌های هوشمند در آن زمان گرانتر

بوده و قابلیت انعطاف‌پذیری کمتری داشتند. اما امروزه سیستم عامل‌ها برنامه‌های مختلف را پشتیبانی می‌کنند.

برای توسعه برنامه‌های کاربردی که نیازمند محیط امن جهت اجرا و عملکرد هستند، کارت‌هایی که رونق بیشتری در محیط‌های تجاری دارند توصیه می شود. از این مجموعه می‌توان به Java Card و MULTOS اشاره کرد.

۶-۵-۱- نسل های مختلف سیستم عامل های کارت های هوشمند

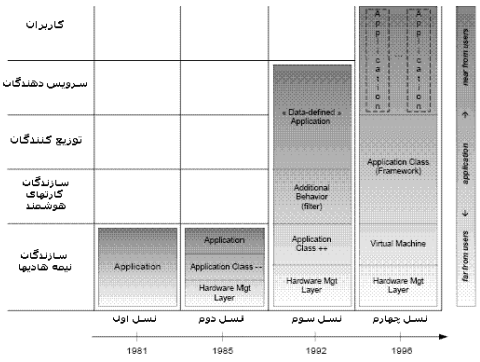
سیستم عامل‌های نسل اول (Monolithic OS) : سیستم عامل‌های یکپارچه‌ای که مجموعه‌ای ترکیب شده از برنامه‌های مورد نیاز مشتری که بطور خاص برای چیپ‌ها و سخت‌افزارهای خاص طراحی شده بود.

سیستم عامل‌های نسل دوم : در این نسل از سیستم عامل‌ها تفکر استفاده مجدد از نرم‌افزارها و moduleها رشد کرد. کارت‌هایی که در دو نسل اول ساخته شده بود. هنگامیکه از محل تولید خارج می شد، فقط در همان مسیر هدف اولیه که طراحی شده بود قابل استفاده بوده و امکان هیچگونه تغییری بعد از ساخت در آن وجود نداشت.

سیستم عامل‌های نسل سوم : در این نسل به دلیل نیاز به استفاده مجدد از روال‌ها و کدهای موجود در پلتفرم‌های مختلف، نرم‌افزارها به بخش‌های مختلف شکسته شد تا قابلیت تطبیق آنها با پلتفرم‌های مختلف افزایش یابد. این تقسیم بندی بطور عام شامل روال‌های کنترلی سخت‌افزار و روال مربوط به نرم‌افزارها و برنامه‌های سرویس دهنده است.

سیستم عامل‌های نسل چهارم : تحولات بوجود آمده بین نسل سوم و چهارم بیشتر مربوط به نحوه سرویس دادن نرم‌افزارها مربوط می‌شد و به این معنی که هنگامیکه نوع سرویس و کاربرد کارت تغییر پیدا کرد، نیازی به صدور کارت‌های جدید با قابلیت جدید نبوده و همان کارت قابلیت تغییر در نرم‌افزار و سرویس را داشته باشد (سیستم عامل‌های پلتفرم باز). نسل اول از کارت‌های هوشمند از سال ۱۹۸۱ ساخت شد و محصولات آن کارت‌های اعتباری و کارت‌های حافظه بودند. ساخت نسل دوم از سال ۱۹۸۵ شروع شد و نمونه محصولات آن کارت‌های بهداشت و سلامت بود. نسل سوم از تاریخ ۱۹۹۲ شروع به تولید شد و محصولات آن کارت‌های CQL و سیم کارت‌ها بود. نسل چهارم هم از سال ۱۹۹۶ شروع به تولید شد که کارت‌های متن باز مانند Java Card و MULTOS از این نسل بودند. با توجه به نمودار تحولات کارت‌های

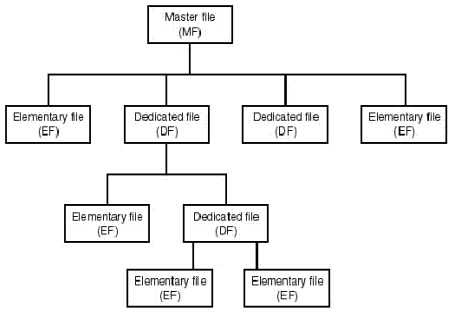
هوشمند شکل(۳-۳)، دو نکته مهم قابل استنتاج است: اول اینکه از دوران کارت‌های یکپارچه تا دوران کارت‌های متن باز، تغییرات در معماری نرم‌افزارها بصورت تدریجی نمایان شده و مجموعه تحولات بصورت مجزا به سیستم‌عامل و برنامه کاربردی تقسیم‌بندی شده و در طی تحولات موجود در نمودار، برنامه‌های کاربردی به لایه کاربران و استفاده کنندگان نزدیکتر شده است. به نظر آرمانی و ایده‌آل است اگر آرزوی یک پلتفرم متن‌باز ماورایی را داشته باشیم که تمامی احتیاجات نرم‌افزاری را از طریق تولید مجموعه‌ای از امکانات جامع توسط برنامه‌سازان، برآورده سازد. این به این معنی است که طراحان نرم‌افزارهای کاربردی، مجبور به طراحی و پیاده‌سازی تمام امکانات مورد نیازخود باتمامی نتایج مضر وزود هنگام آن باشند.



شکل ۶- ۳: سیر تحولات معماری کارت‌های هوشمند

۶-۵-۲- فایل سیستم‌های کارت‌های هوشمند

اغلب سیستم عامل‌های کارت‌های هوشمند یک فایل سیستم بر پایه استاندارد ISO 7816 را پشتیبانی می‌کنند. کارت هوشمند دارای جزء بیرونی نمی‌باشد، لذا فایل سیستم آن در واقع یک جزء نزدیک به حافظه کارت است. فایل سیستم کارت دارای ساختار سلسه مراتبی است که در آن فایل‌ها می‌توانند دارای نام‌های الفبایی طولانی نام‌های عددی کوتاه و نام‌های مرکب باشند.



شکل۶-۴: فایل سیستم سلسله مراتبی کارت

سیستم عامل‌های کارت‌های هوشمند اعمال اصلی فایل از جمله ایجاد، حذف، خواندن، نوشتن و به‌روزرسانی را پشتیبانی می‌کنند. علاوه بر آن اعمال خاصی را بسته به نوع فایل نیز شامل می‌شوند. برای مثال فایل‌های خطی، سری‌های متعددی از رکوردهای با طول ثابت را شامل می‌شوند که می‌توانند از طریق شماره رکورد و یا خواندن متوالی رکوردهای بعدی و قبلی در دسترس باشند. فایل‌های دایره‌ای نیز همان فایل‌های خطی هستند که پس از خواندن یا نوشتن رکورد آخر، به رکورد اول منتهی می‌شوند. فایل سیستم‌های purse و transparent نیز انواع دیگری از فایل‌های کارت‌های هوشمند می‌باشند.

همراه با هر نوع فایل سیستم استفاده شده در کارت یک لیست سطح دسترسی نیز وجود دارد که دسترسی‌های هر ID را تعیین می‌کند. برای مثال B برای خواندن یک فایل خاص دسترسی دارد، ولی مجاز به update آن نمی‌باشد.

۶-۵-۳- واحد انتقال داده(Application Protocol Data Units (APDUs

واحد اصلی تبادل اطلاعات با کارت هوشمند APDU است. پیام ارسال شده از لایه نرم‌افزار کاربردی و پاسخ داده شده به آن، واحد انتقال داده (APDU) نامیده می‌شود. ارتباط بین کارت و دستگاه خواننده از طریق APDU برقرار می‌شود. APDU در واقع یک بسته داده‌ای است که دستورات کامل و یا پاسخ‌های کامل را در بر دارد.

استاندارد ISO 7816-4 دو نوع APDU را تشریح می‌کند: APDUهای دستوری؛ که به کارت فرستاده می‌شوند؛ و APDUهای پاسخی، که از کارت برای پاسخ به دستورات، فرستاده می‌شوند.

APDU ها از فیلدهای زیر تشکیل می‌شوند:

قالب APDUهای دستوری:

CLA	INS	P1	P2	Lc	Data	Le

هر APDUی دستوری شامل بخش‌های زیر است:

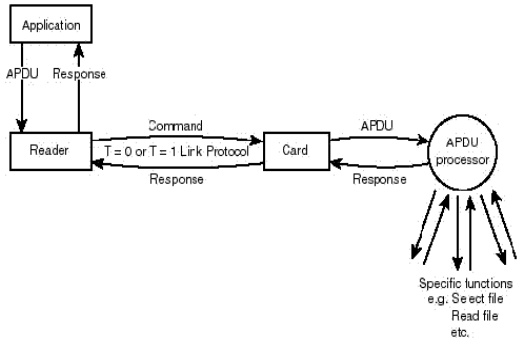
- یک نوع کلاسی (CLA)؛ این نوع کلاس دستورات را مشخص می‌کند. برای مثال اگر دستورات از نوع ISO باشند و یا اینکه از روش Secure messaging پیام‌دهی ایمن، استفاده کرده‌اند.
- یک بایت دستورات (INS)؛ این بایت مشخص کننده دستور است.
- دو بایت متغیر P1 و P2. این دو بایت برای ارسال متغیرهای دستورات استفاده می‌شوند.
- یک بایت طول (Lc-Length command)؛ این بایت مشخص کننده طول داده ارسال شده به کارت از طریق همین APDU می‌باشد.
- داده اختیاری، این قسمت برای ارسال داده‌های واقعی به کارت برای پردازش می‌باشد.
- یک بایت طول (Le-Length expected)؛ این بایت مشخص کننده طول داده‌ای است که در APDUی متوالی پاسخ برگردانده می‌شود.

قالب APDUهای پاسخی:

Data	SW1	SW2

هر APDU ای پاسخی شامل بخش‌های زیر است:

- داده اختیاری
- دو بایت وضعیت SW1 و SW2؛ این بایت‌ها شامل اطلاعات وضعیت است که در استاندارد ISO 7816-4 آمده است.
- نرم‌افزار برنامه کاربردی از یک پروتکل بر پایه APDU استفاده می‌کند تا تبادل اطلاعات بین کارت و خواننده را انجام دهد. یک جزء از نرم‌افزار کارت این APDUها را تشریح کرده و دستورات آن را اجرا می‌کند. این معماری در شکل (۶-۵) مشخص شده است.



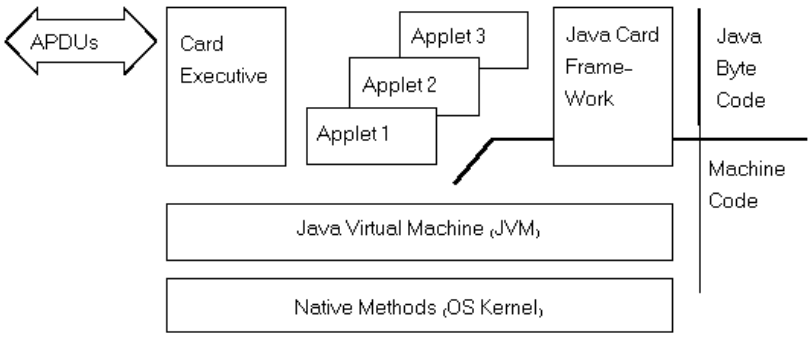
شکل ۶-۵: معماری ارتباط با برنامه کاربردی

۶-۶- سیستم عامل های کارت های چند برنامه

سیستم عامل‌های چند برنامه امکان توسعه و ساخت چندین برنامه را بر روی یک کارت به توسعه دهندگان می دهد. از این مجموعه می‌توان به Java Card و MULTOS و WINDOWS Card اشاره کرد.

۶-۶-۱- جاوا کارتJava Card

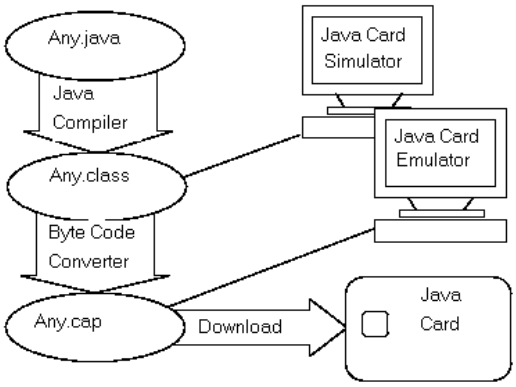
کارت‌های جاوا به وسیله Schlumberger معرفی شد و اخیراً به عنوان یک استاندارد توسط JavaSoft پذیرفته شد. کارت هوشمندی با پتانسیل پذیرش استانداردهای کارت هوشمند؛ دربرداشتن انواع APIهایی که به Appletهای جاوا اجازه می‌دهد تا به طور مستقیم در استاندارد ISO 7816 اجرا شوند؛ از مشخصه‌های اصلی یک کارت جاوا می‌باشد. کارت‌های جاوا اجرای ایمن برنامه‌های کاربردی مختلف را تضمین می‌کنند. معماری سیستم عامل کارت جاوا در شکل (۶-۶)قابل مشاهده است.



شکل۶-۶ : معماری سیستم عامل کارت Java

سیستم عامل کارت جاوا به برنامه‌های کاربردی روی کارت اجازه می‌دهد تا به زبان جاوا نوشته شوند. این ویژگی به ما امکان می‌دهد تا نرم‌افزارهای روی کارت را مستقل از جاوا توسعه دهیم و این امکان برای سازندگان سیستم عامل‌های کارت از اهمیت خاصی برخوردار است. علاوه بر این کارت‌های جاوا به عنوان پایه‌گذاری برای کارت‌های چندبرنامه‌ای به شمار می‌آیند و می‌توانند در یک زمان اجرای چند برنامه کاربردی را پشتیبانی کنند. برنامه‌های اجرایی روی کارت به عنوان Appletهای کارت هستند که شامل بایت کدهای خاص جاوا می‌باشند و در Java Card Runtime Environment تشریح می‌شوند.

فرآیند توسعه یک Applet کارت به شرح شکل(۶-۷) است:



شکل۶-۷: فرآیند توسعه یک Card Applet

جاوا کارت تغییراتی در معماری کارت‌های هوشمند هم برای تولید کننده و هم توسعه دهندگان اعمال کرد. جاوا کارت فراهم آورنده تسهیلات بیشتر و انعطاف‌پذیری زیادی برای کاربران در حین رد و بدل اطلاعات بوده و فرصت‌های زیادی برای توسعه دهندگان در تمامی ابعاد تجاری، ایجاد نمود.

جاوا کارت امکان اجرای متغیر برنامه‌ها را فراهم آورده است. به این معنی که یک کارت هوشمند با سیستم عامل جاوا کارت روی آن، قابلیت تغییر تنظیمات و ویژگی‌ها را در طول اجرای برنامه‌ها دارد. به عنوان مثال یک کارت هوشمند در زمانی می تواند نقش یک کیف پول الکترونیکی را داشته و برای مصارف بانکی استفاده شده و در زمان دیگری بعنوان کارت پرسنلی جهت احراز هویت شخصی استفاده گردد. این از قابلیت های فوق العاده برای توسعه دهدگان و مصرف کنندگان است.

MULTOS–۲–۶–۶

MULTOS و یا در واقع MULT-OS توسط شرکت بین‌المللی Mondex ارائه شد و یک سیستم‌عامل متن-باز بسیار امن چند برنامه‌ای برای کارت‌های هوشمند است. این سیستم عامل اجازه می‌دهد تا در یک زمان چند برنامه در فضایی کاملاً امن اجرا شوند. شرکت Mondex برای برنامه نویسی کاربردی یک زبان بهینه کارت هوشمند را با نام MEL (MULTOS Enabling Language) و MULTOS-API عرضه کرده است. ویژگی MULTOS این است که متن باز بوده و توسط سازمان‌های بین المللی با عنوان کنسرسیوم MAOSCO که توسعه و گسترش کارت هوشمند را در بین تمامی تکنولوژیهای موجود در دنیا بر عهده دارد؛ هدایت می‌شود. MULTOS توسط شرکت MAOSCO گواهی (Licensed) شده است. (MAOSCO, 2000)

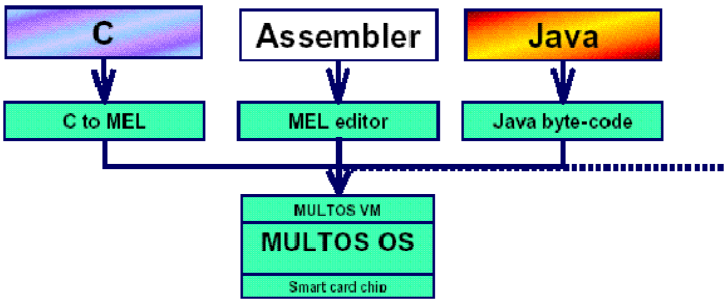
مهمترین ویژگی MULTOS استقلال آن از زبان‌های برنامه نویسی است:

زبان برنامه نویسی اسمبلی: MULTOS تنها محیطی است که دارای زبان اسمبلی ساده‌ای می‌باشد. برنامه‌های کارت‌های MULTOS عموماً به زبان MEL(MULTOS Executable Language) نوشته می‌شوند که شامل دستورات اسمبلی به علاوه functionهای کارت با نام functionهای اولیه است.

زبان برنامه‌نویسی C: MULTOS تنها محیطی است که در حال حاضر دارای کامپایلر C است. ابزار برنامه نویسی SwiftC متعلق به SwiftCard یک کامپایلر ANSI است که به شما اجازه می‌دهد برنامه‌های موجود را به سرعت در سیستم عامل MULTOS اجرا کنید.

زبان برنامه نویسی Java: هر دوی سیستم عامل‌های JavaCard و MULTOS زبان برنامه نویسی جاوا را پشتیبانی می‌کنند. در هر دو حالت یک کامپایلر جاوا برنامه را به کلاس‌های جاوا ترجمه می‌کند. برای JavaCard کلاس‌ها به بایت کدهای جاوا ترجمه شده و در MULTOS کامپایلر SwiftJ کلاس‌های جاوا را به کدهای MEL ترجمه می‌نماید.

برنامه‌نویسی Visual Basic: کارت‌های هوشمند برای Windows این زبان برنامه نویسی را برای توسعه نرم‌افزار انتخاب کرده‌اند. برای توانمند ساختن MULTOS در اجرای برنامه‌های VB، تکنولوژی SwiftCard هم اکنون در حال کار بر روی مترجم VB به MEL است.



شکل۶-۸: معماری چندبرنامه ای MULTOS

با توجه به این ویژگی‌ها، MULTOS چیزی فراتر از یک سیستم عامل است. MULTOS یک شمای کامل برای مدیریت برنامه‌های کاربردی کارت‌های هوشمند است. این یک فرآیند ایمن، کافی و مقرون به صرفه برای مدیریت برنامه‌ها در دنیای جدیدی است که در آن یک کارت ممکن است برنامه‌های مختلفی با زبان‌های مختلف را در خود جای داده باشد.

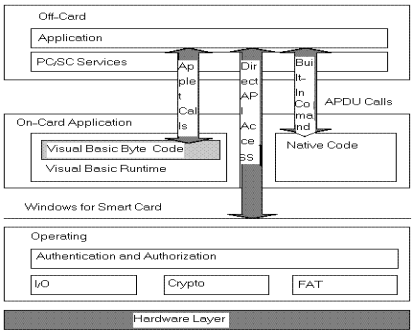
MULTOS در سطح بالایی از امنیت به تولید کنندگان و برنامه نویسان و سایر کاربران خود تضمین می‌دهد که برای رسیدن به اهداف خود نیاز به تقبل ارزیابی‌های پرهزینه و طولانی مدت ندارند.

۶-۶-۳- کارت‌های Windows

در سال ۱۹۹۹ شرکت مایکروسافت وارد دنیای کارت هوشمند شد و نسخه جدیدی از Windows را برای کارت هوشمند به عنوان سیستم عامل عرضه کرد. به عنوان جدیدترین عضو خانواده سیسیتم عامل‌های Windows، این Windows مزایای محیط Windows را با خود برای کارت‌های هوشمند به ارمغان آورد.

Windows کارت‌های هوشمند، یک سیستم عامل ۸ بیتی چند برنامه‌ای است که با یک ROM ۸کیلوبایتی کار می‌کند. این سیستم عامل، کم هزینه، و ساده برای برنامه‌نویسی به زبان‌های ویژوالی مانند VB است؛ و برای ورود کامپیوترهای شخصی به دنیای کارت هوشمند طراحی شده است.

معماری Windows برای کارت‌های هوشمند در شکل(۶-۹) قابل مشاهده است.



شکل۶-۹: معماری Windows برای کارت‌های هوشمند

همانند Java Card، توسعه برنامه‌هایی که در کارت با سیستم عامل‌های Windows اجرا می‌شوند؛ زبان‌های سطح بالای برنامه‌نویسی مطرح می‌شوند. شرکت مایکروسافت به جای جاوا، از بایت کدهای ساخته شده توسط برنامه‌های VB که در کارت اجرا می‌شوند، استفاده می‌کند. Appletهای روی کارت با Appletهای بیرون کارت از طریق APDUهای معمولی ارتباط برقرار می‌کنند. سیستم عامل، یک API برای کار با محتویات کارت ارائه می‌دهد. این API با یک زبان بی طرف طراحی شده و می‌تواند به وسیله VB یا appletهای اصلی قابل دسترسی باشد.

۶-۷- نتایج

نکته مهم در مورد کارت‌های هوشمند این است که این کارت‌ها ابزاری است که هر روزه توسط افراد جامعه به کار گرفته می‌شوند. این کارت‌ها حاوی اطلاعات مهمی است که به صورت الکترونیکی ذخیره شده‌اند.

هوشمندی کارت‌های هوشمند از مدارهای مجتمعی که در داخل کارت پلاستیکی قرار گرفته‌اند؛ نشات می‌گیرد. این قابلیت می‌تواند با قرار گرفتن این مدارها در ابزارهای دیگر مانند ساعت، عینک، حلقه و یا گوشواره نیز قرار گیرند.

کارت‌های هوشمند تکنولوژی جدیدی هستند که زندگی روزمره مردم را تحت تاثیر قرار داده‌اند. نمونه‌هایی از تاثیر این تکنولوژی نوین در زندگی روزمره را می‌توان در عملیات بانکی، خرید، ویزیت پزشکان و تلفن مشاهده نمود.

مطالعات و تحقیقات انجام شده دز زمینه کارت‌های هوشمند و نرم‌افزارهای آنان دو نکته مهم را بعنوان تمهیدات تولید کارت هوشمند مطرح می کند :

۱- کارت‌های هوشمند می‌بایست در سطوح بسیار پائین در اختیار طراحان برنامه‌های آنان قرار گیرد. جهت جلوگیری از رکود کارایی و سرعت در زمانیکه اجزای مختلف با یکدیگر تلفیق می‌گردند، سیستم عامل می‌بایست در بالاترین سطح قرار گرفته و سرویس دهندگان می‌بایست اجزا برنامه‌های مختلف را نزدیکتر به سخت‌افزار طراحی نماید. به عبارت دیگر سیستم عامل می‌بایست قابلیت بسط و توسعه و تعمیم یافتن را دارا باشد.

۲- کارت‌های هوشمند می‌بایست به طراحی نوین و مدرن سیستم‌های مکانیزه توزیعی، نزدیکتر شود. تحولات پلتفرم‌های کارت‌های هوشمند نباید مشتق از نیازهای امروزی کاربران باشد، بلکه باید با دید به آینده فکر کرده و حرکت کرد. کارت‌های هوشمند باید به عنوان پلتفرم‌های اجرای چند برنامه‌ای تبدیل شوند و نحوه طراحی آنها باید به گونه‌ای باشد که تمامی ابعاد نیازهای موجود در نرم‌افزار در آن دیده شود.

باتوجه به مطالب بیان شده احساس می‌گردد که نظر به روند فناوری اطلاعات درجهان و به منظور جلوگیری از عقب ماندن از این رشد، کشور ما نیز می‌بایستی خودرا دراین روند سهیم کرده و با کشورهای دیگر در این جهت همگام شود.یکی از ابزارهایی که در انتقال اطلاعات مورداستفاده قرار می گیرد، کارت‌های هوشمند است. این کارت‌ها دارای فوایدی هستند ازقبیل کاهش زمان و هزینه، انتقال بهتر و سریعتر اطلاعات و در کل بهبود انتقال اطلاعات بر این

اساس بایستی مسئولان محترم با درنظر داشتن شرایط مختلف، امکان استفاده از این ابزارها را در کشور فراهم آورده و راهکارهای استفاده از آن را ایجاد کنند. ما نیز به عنوان افراد این جامعه بایستی خود را در این تکنولوژی سهیم دانسته و در انتقال و پیاده‌سازی فرهنگ لازم و استفاده از آن، دولت را یاری کنیم تا زمان پیاده‌سازی چنین تکنولوژی هرچه کوتاهتر گردد.

فصل هفتم – بررسی و مقایسه سیستم عامل‌های کارت هوشمند متن بازموجود

تکنولوژی کارت‌های چند برنامه‌ای در حال حاضر به نقطه‌ای از تکامل و پیشرفت رسیده اند که بطور گسترده‌ای در بسترهای گوناگون همانند گوشی‌های موبایل، سیستم‌های بانکی، سیستم‌های احراز هویت و اطلاعات پرسنلی و امنیتی استفاده می‌شوند. در حال حاضر برجسته‌ترین پلتفرم‌ها و سکوهای موجود در این کارت‌های هوشمند، Multos و JavaCard هستند. هر دو پلتفرم دارای معماری یکسان و مشابهی در هسته خود و دارای یک ماشین مجازی جهت تفسیر کدهای اجرایی برنامه‌ها می‌باشند. این سناریو امکان اجرای برنامه‌ها را بدون تغییر در سخت‌افزارهای مختلف فراهم می‌آورد. به منظور تشریح علل موفقیت سیستم عامل جاوا کارت و Multos در بین سیستم عامل‌های متن باز موجود برای کارت‌های هوشمند، ویژگی‌های تحقق یافته آنان را بررسی می‌کنیم. این سیستم عامل‌ها با افزودن قابلیت‌های جدید می‌توانند موقعیتی بسیار بهتر از امروز را دارا باشند. لازم به ذکر است که هیچ تکنولوژی بدون داشتن کاربران و شرایط محیطی درست، نمی تواند موفق باشد. برای نمایش شباهت‌ها و تفاوت‌های پلتفرم‌های Multos و JavaCard متن زیر توصیف کوتاه و اجمالی از ویژگی‌های Multos و JavaCard است.

۷–۱– بررسی سیستم عامل MULTOS

Multos امن ترین سیستم عامل متن باز کارت هوشمند بوده که دارای خاصیت چند برنامگی می باشد. پلتفرم Multos امکان اجرای چندین برنامه مختلف را در کنار هم و بصورت مجزا و با امنیت بالا در یک کارت هوشمند فراهم می‌آورد. Multos توسط کنسرسیومی از چندین شرکت با نام Maosco پشتیبانی و کنترل می‌شود. این اتحادیه دارای عضویت آزاد و رایگان میباشد.

۷–۱–۱– قابلیت‌ها و مزایای Multos

امنیت بالا. Multos بستر اصلی ایجاد MasterCard بوده و دارای امنیت بالا می‌باشد و تنها سیستم عامل کارت هوشمند است که دارای مقام E6 در رده بندی امنیت و استاندارد ITSEC می‌باشد. طراحی منحصر بفرد آن امکان بارگذاری و حذف نرم‌افزارهای کاربردی آنرا با امنیت بالا فراهم می‌آورد.

پایداری عملکرد. نسخه‌های مختلف Multos تحت سخت ترین آزمایش‌ها قرار میگیرند تا اطمینان حاصل شود که Multos پلتفرم پایداری برای کارت هوشمند می‌باشد. این بدان معنی است که نرم‌افزارهای مشابه می‌توانند بر روی تمامی نسخه‌های مختلف Multos اجرا شوند و نیاز به هیچ تغییری نداشته باشند. صرفنظر از اینکه کارت با کدام سیستم مدیریت شده و چه سخت‌افزار با کدام Chip در آن وجود دارد.

انعطاف پذیری. Multos طیف وسیعی از زبان‌های برنامه نویسی را برای توسعه و پیاده سازی پشتیبانی می‌کند. تولید کنندگان نرم‌افزار، محدود به یک زبان و یک ابزار برای توسعه برنامه‌ها و تست آنها نیستند.

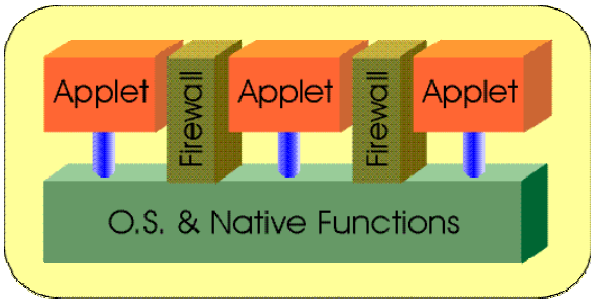
انتشار سریع. قوانین کاربرد و انتشار Multos بنحوی طراحی شده اند که Multos بصورت ساده تر و سریعتر از دیگر رقبا گسترده و منتشر شوند. هم اکنون بیش از ۱۶ میلیون کارت Multos توسط توزیع کنندگان کارت‌های Multos در سراسر دنیا پخش شده است. امروزه بسیاری از سازمان‌های دولتی در کشورهای مختلف رو به پلتفرم Multos آورده و از مزایای امنیت اطلاعات در آن استفاده می‌کنند. به عنوان مثال کشور هنگ کنگ Multos را به عنوان سیستم عامل مورد استفاده در کارت‌های ملی – هویتی انتخاب کرده است.

Multos بطور خاص برای کارت‌های با منابع و ملزومات کم و امنیت بالا طراحی شده است. بطوریکه توزیع کنندگان بعنوان یک پلتفرم مطمئن و امن برای بارگذاری و حذف برنامه‌های کاربردی، بر روی آن متمرکز شده اند. دیگر سیستم عامل‌های موجود حاصل تغییرات اعمال شده در دیگر سیستم‌های کامپیوتری مرسوم می‌باشد که اهداف و توجه خاصی به نحوه بارگذاری و اجرای برنامه‌ها در کارت ندارند.

کارت‌های Multos معمولاً از سخت‌افزارها و Chipهای رایج که دیگر انواع کارت‌ها از آن استفاده می‌کنند پشتیبانی می‌کند. کارت‌های Multos بر روی کارت‌های با حافظه پائین EEPROM هم نصب و اجرا می‌شود. این عامل تاثیر به سزایی در تعیین قیمت کارت دارد. از دیدگاه دیگر کارت‌های Multos که دارای قابلیت اجرای چند برنامه هستند از نظر کاربرد ارزش بیشتری از دیگر کارت‌ها دارند که باعث افزایش نسبی قیمت کارت‌های Multos در این مورد می‌باشد.

هسته اصلی Multos دارای یک مفسر است که امکان ساخت برنامه‌های مختلف را فارغ از سخت‌افزار کارت، فراهم می‌آورد. با مجموعه API موجود در کارت، برنامه‌هایی که از این API

برای ساخت استفاده کنند یکبار نوشته شده و در همه جا بر روی هر نسخه‌ای از Multos اجرا می‌شود.



شکل۷-۱: ساختار داخلی MULTOS

باتوجه به گواهینامه امنیتی ITSEC E6 ، Multos تفکیک پذیری برنامه‌ها را مدیریت و کنترل می‌کند. گواهینامه ITSEC E6 بالاترین رده امنیتی در استانداردهای ITSEC است. از اینرو می‌توان Multos را بعنوان امن ترین کارت هوشمند در میان رقبای خودش معرفی کرد.در شکل(۷-۱) ساختار داخلی Multos که شباهت زیادی به جاوا کارت دارد نمایش داده شده است. قابل ذکر است که پیاده سازی و عملکرد داخلی آن کاملا متفاوت است.Multos از یک زبان برنامه سازی خاص خودش به نام MEL (Multos Executable Language) استفاده می‌کند که یک زبان ساده مجازی برای پردازنده آن است. برنامه سازان می‌توانند برنامه‌های خود را با یک زبان سطح بالا مانند C نوشته و از طریق ابزار ترجمه به مفسر زبان MEL ترجمه می‌کنند و سپس کدهای ترجمه شده به کارت منتقل می شوند.

سیستم عامل کارت‌های Multos نیز بصورت سیستم عامل‌های چند برنامه‌ای طراحی شده که می تواند یک یا چند برنامه نوشته شده به زبان‌های سطح بالا را در خود بپذیرد. از آنجا که Multos در ابتدا توسط موسسه‌های مالی بعنوان کیف پول الکترونیکی طراحی شده بود، امنیت کارت‌های هوشمند بعنوان یک شاخص مهم در طراحی آنها مطرح است. از آنجائیکه Multos برای ذخیره ساختارهای سازماندهی نشده طراحی شده است، بسیاری از موسسه‌های مالی و بانک‌ها تمایل دارند که از آنها در سیستم‌ها و برنامه‌های خود استفاده نمایند. بوسیله کمک پردازنده رمزنگار در کارت، عملیات الگوریتم‌های رمزنگاری از قبیل DES و RSA بسیار سریعتر انجام خواهد شد.

۷-۱-۲- انواع برنامه هایMultos (برنامه‌های پوسته/برنامه‌های استاندارد)

در Multos یک برنامه، یا از مجموعه برنامه‌های استاندارد کارت بوده و یا از برنامه‌های پوسته کارت می‌باشد.

برنامه‌های استاندارد، برنامه‌های نرمال و عادی در کارت می‌باشند که هر یک از آنها نیاز به یک فایل فعال در کارت دارند و در صورتیکه یک برنامه پوسته در حال اجرا نباشد، بطور پیش فرض عملکرد کارت را کنترل می‌کند.

برنامه‌های پوسته همانند برنامه‌های استاندارد می‌باشند با این تفاوت که بصورت ضمنی در هنگام اجرا جایگزین فایل اصلی در ساختار فایل‌های کارت می‌گردد. به همین علت در هر لحظه فقط یک برنامه پوسته‌ای می‌تواند در حال اجرا و کنترل دیگر برنامه‌ها بوده و همیشه می‌بایست زودتر از دیگر برنامه‌ها اجرا گردد.

برنامه‌ها بصورت بصورت پوشه‌ها و دایرکتوری‌هایی مطرح شده اند که حاوی فایل‌های مقدماتی برنامه می‌باشند. فضای اطلاعاتی برنامه بعنوان فضای مخصوص برنامه جهت نگهداری اطلاعات مورد نیاز برنامه تلقی می‌شود. بطور پیش فرض تمامی اطلاعات از دید و دسترسی دیگر برنامه‌ها و ترمینال‌های دیگر پنهان می‌باشد.

همانطور که مطرح شد استاندارد ۷۸۱۶ از فایل‌های خاص بصورت دایرکتوری و پوشه استفاده می‌کند. همچنین استاندارد ۷۸۱۶ ارائه کننده دستوراتی برای فعال کردن فایل‌های خاص جهت انتخاب شدن و اجرا شدن می‌باشد. برای انتخاب یک زیر پوشه (Sub Directory) Multos تمامی دستورات مخصوص برنامه‌ها را به برنامه انتخاب شده هدایت می‌کند. در این روال هنگامیکه یک برنامه انتخاب شد، با ارسال دستورات بعدی آماده اجرا شده و سپس اجرا می‌گردد. برای اجرای دستورات فراخوانی شده می‌بایست یکی از برنامه‌های روی کارت انتخاب شده باشد. برنامه‌ها با دریافت دستورات ارسال شده با توجه به نوع مکانیزم عملکرد خود، دستورات را پردازش و اجرا می‌کنند.

۷-۱-۳- شاخص برنامه‌ها

شاخص برنامه‌ها یک شماره بین‌المللی بوده که توسط استاندارد ISO 7816 تعیین شده و بطور منحصربفرد مشخص کننده یک برنامه خاص و واحد در کارت منتشر شده است. هر سیستم عامل کارت هوشمندی که از این استاندارد پیروی کند، دارای AIDهای منحصر بفرد

است و Multos نیز با پیروی از استاندارد ISO 7816 اطمینان داده است که شاخص‌های برنامه‌ها منحصر بفرد بوده و هیچ دو برنامه‌ای با یک شاخص اجرا نشوند. این شاخص به منظور انتخاب برنامه‌ها در کارت است .

۷-۱-۴- کنترل کننده مشخصه فایل

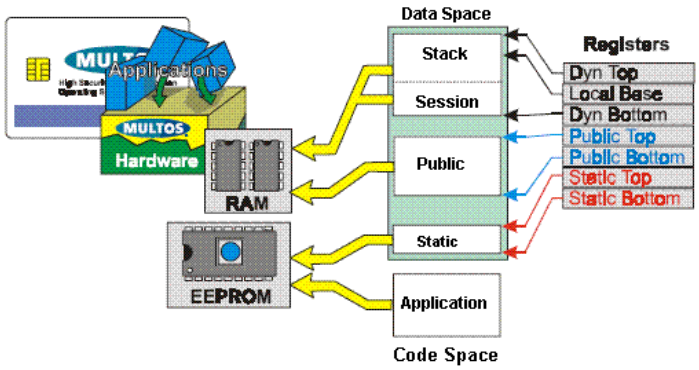
کنترلر مشخصه فایل، واحدی در کارت است که امکان دریافت مشخصات و ویژگی‌های فایل‌های روی کارت را از طریق دستگاه‌های رابط و یا کارت‌خوان‌ها فراهم می‌آورد. اطلاعات ویژگی‌های فایل‌ها مجموعه‌ای از شاخص‌هایی است که توسط استاندارد ISO 7816 جهت تبادل اطلاعات فایل تبیین شده است و Multos نیز از این امر مستثنی نیست.

۷-۱-۵- معماری حافظه

در Multos هر برنامه شامل مجموعه‌ای از قطعات و بخش‌های مختلف می‌باشد که در نگهداری و اجرای برنامه شرکت می‌کنند و بطور مجزا از فضای تشکیل دهنده برنامه‌های دیگر می‌باشند. این مجموعه به ماشین انتزاعی برنامه‌ها در Multos نامیده می شود (Multos Application Aobstract Machine AAM).

AAM برای هریک از برنامه‌های Multos فضای مجزایی برای اطلاعات و کدهای آنها معین می‌کند و هر برنامه در Multos دارای یک نقشه حافظه با همان ساختار کنترلی AAM می‌باشد. برای نمونه هر برنامه دارای یک فضای اختصاصی ثابت بوده که در فضایی در انتهای فضای اطلاعات برنامه قرار میگیرد.

برنامه‌ها اجازه مشاهده و دسترسی به فضای حافظه دیگر برنامه‌های را ندارند. تنها فضایی که همه برنامه‌ها می‌توانند آنرا ببینند و به آن دسترسی دارند فضای عمومی در حافظه کارت است و Multos در میان فضاهای دیگر که اختصاصی می‌باشند دیواره آتش قدرتمندی قرار داده است. AAM اجازه نمی‌دهد که یک برنامه در فضای اختصاصی برنامه دیگری بصورت نوشتنی / خواندنی دسترسی داشته باشد.



شکل ۷-۲: معماری حافظه MULTOS

آدرس محلی از حافظه در فضای داده‌ها (Data Space) که توسط AAM در اختیار برنامه‌ها قرار میگیرد با آدرس واقعی فیزیکی که سخت‌افزاری تعیین شده است متفاوت است. بعنوان مثال فضای کدها برای یک برنامه که توسط AAM تخصیص داده می‌شود از صفر شروع شده و برای تمامی برنامه‌های استفاده می‌گردد و این نحوه آدرس دهی به آدرس‌های فیزیکی سخت‌افزاری ارتباطی ندارد.

AAM دو فضای مجزای حافظه تعیین کرده است، یکی برای کدهای برنامه‌ها و دیگری برای داده‌ها و اطلاعات:

- فضای کدها : فضای کد اشاره به قسمتی از حاقظه دارد که توسط کدهای برنامه‌ها اشغال شده است. این فضا قابل نوشتن توسط برنامه‌ها نیست و فقط قابلیت اجرا دارد.
- فضای داده‌ها : این فضا اشاره به قسمتی از حافظه دارد که شامل اطلاعات و داده‌های برنامه‌ها می‌باشد.

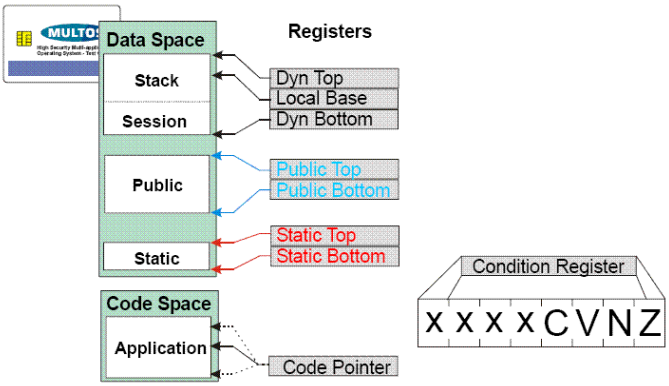
این دو فضا از یکدیگر مجزا بوده و توسط رجیسترهای مختلفی آدرس دهی می‌شوند.

۷-۱-۶- معماری ثبات‌ها

واحد AAM در Multos دارای نه ثبات (رجیستر) می‌باشد. هفت ثبات از نه ثبات موجود در دسترس و اختیار برنامه نویسان قرار گرفته که با مکانیزم آدرس دهی بصورت افست توانایی دسترسی و آدرس دهی فضای داده‌ها (Data Space) را به جای آدرس دهی مستقیم از طریق

سگمنت‌های حافظه، دارند. دو ثبات دیگر برای نگهداری اشاره گر کد (Code Pointer) و کدهای شرطی (Condition Register) در نظر گرفته شده اند.

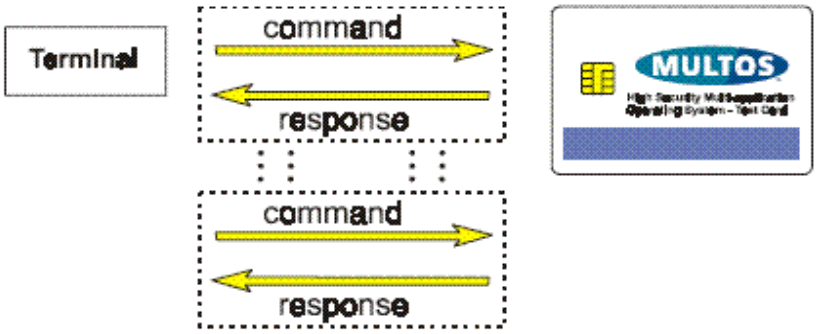
بر خلاف دیگر زبان‌های اسمبلی، MEL دارای ثبات داده‌ها نمی‌باشند. ثبات‌های داده برای نگهداری داده‌های در نظر گرفته شده اند اما ثبات‌های MEL فقط برای نگهداری اشاره گرهای آدرس در نظر گرفته شده اند. شکل(۷-۳) مجموعه ثبات‌ها و رجیسترهای موجود در Multos را که بوسیله AAM فراهم شده اند را نشان می‌دهد.



شکل ۷-۳: ثبات‌ها و رجیسترهای MULTOS

۷-۱-۷- معماری ورودی و خروجی

معماری ورودی و خروجی در کارت هوشمند بر اساس جفت پیام‌های پرسش و پاسخ می‌باشد. ترمینال یا دستگاه کارت خوان دستورالعمل خود را به کارت ارسال کرده و کارت در پاسخ دستورالعمل را پردازش می‌کند و پیام مناسب را برای ترمینال می‌فرستد. همواره تمامی ارتباطات در ورودی و خروجی کارت از طرف ترمینال‌ها شروع می‌گردد. بدین معنی که کارت هیچگاه دستورالعملی برای اجرا و شروع ارتباط به ترمینال ارسال نمی‌کند.



شکل۷-۴: معماری ورودی/خروجی MULTO

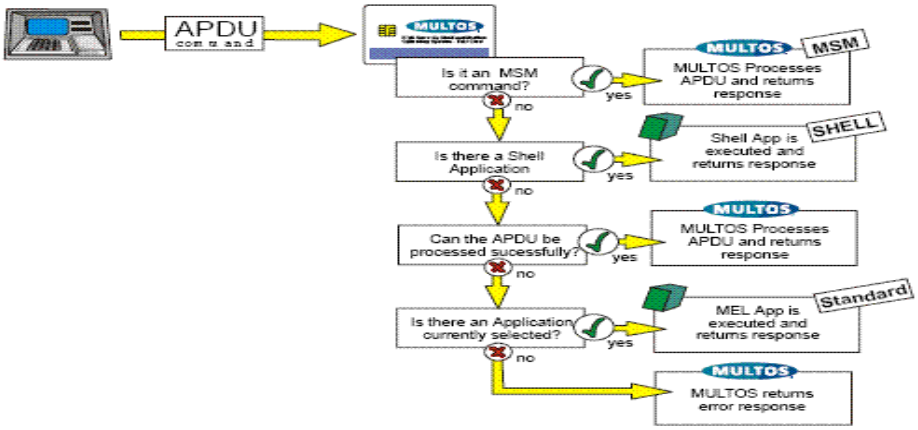
در زیر مراحل دریافت دستورالعمل از ترمینال به کارت بررسی شده و چگونگی پردازش دستورالعمل و ارسال پاسخ نشان داده می‌شود:

هدایتگر دستورالعمل (Command Routing) : این واحد معین کننده محلی است که دستورالعمل بایستی به آنجا رفته و پردازش گردد. به عبارت دیگر از طریق این واحد برنامه مورد نظر برای کار با دستورالغمل ارسال شده مشخص می‌گردد.

اجرای برنامه : این واحد معین می‌کند که هنگامیکه یک APDU دستورالعمل به یک برنامه رسید، چه عملی بر روی آن انجام می‌گردد. برنامه می‌بایست دستورالعمل را خوانده و پردازش کند و پاسخ را از طریق Multos برگرداند.

خاتمه دهنده برنامه : این بخش نمایانگر اتفاقاتی است که پس از پردازش دستورالعمل و ارسال پاسخ به ترمینال برای در برنامه رخ می‌دهد. در این ارتباط Multos عهده دار تمامی ارتباطات سطح پایین سخت‌افزاری می‌باشد.

هنگامیکه Multos دستورالعملی را دریافت می‌کند ، می‌بایست مشخص گردد که این دستورالعمل به سمت کدام برنامه برای‌پردازش و تهیه پاسخ مناسب هدایت گردد.شکل (۷-۵) نمایان کننده این چرخه و روال می‌باشد.



شکل۷-۵: چرخه هدایت دستورالعمل

دستورالعمل‌های امنیتی در Multos همواره وجود داشته و توسط خود سیستم عامل پردازش می‌شوند. در جدول(۷-۱) لیست دستورالعمل‌های کنترلر امنیت در Multos مطرح شده است.

جدول۷-۱: دستورالعمل‌های کنترلر امنیت در MULT

فعال کننده کارت	Load MSM Ciphertext
بارگذاری و اجرای برنامه ها	Open MEL Application
	Load Code
	Load Data
	Load DIR Record
	Load FCI Record
	Load Application Signature
	Load KTU Ciphertext
	Create MEL Application
حذف کننده برنامه های	Delete MEL Application

اگر در کارت برنامه پوسته‌ای در حال اجرا باشد ، دستورالعمل ارسال شده به سمت آن برنامه هدایت می شود. این مورد یکی از اصلی ترین تفاوتهای برنامه‌های پوسته‌ای و برنامه‌های استاندارد می‌باشد. یک برنامه پوسته‌ای تمام دستورات به جز دستورات امنیتی که در بالا ذکر شد را دریافت می‌کند .

اگر به هنگام ارسال دستورالعمل هیچ برنامه پوسته‌ای در حال اجرا نباشد ، Multos دستورالعمل وارد شده را بررسی کرده و مشخص می‌کند که آیا این دستورالعمل از دستورات استاندارد ۷۸۱۶ می‌باشد یا نه.

اگر به هنگام ارسال دستورالعمل برنامه استانداردی در حال اجرا باشد ، آن دستورالعمل به سمت برنامه در حال اجرا هدایت می‌گردد. واحد APDU مربوط به دستورالعمل در فضای عمومی حافظه نوشته شده و سپس اجرا می‌گردد.

۷-۱-۸- زبان اجرایی Multos

زبان اجرایی Multos توسط سیستم عامل آن تفسیر و اجرا می‌گردد. تمامی برنامه‌های Multos در نهایت بعنوان یک واحد حافظه MEL اجرا می‌شوند ، خواه با زبان اسمبلی نوشته شده باشند و یا با هر زبان سطح بالای دیگری. در این سلسله مقالات Multos بعنوان سیستم عامل در نظر گرفته شده است، در صورتیکه از دید وسیعتر و عمیقتر Multos تمامی فرایند ساخت و تولید ، انتشار، بازسازی و امنیت را بر عهده دارد. MEL یک زبان است. این زبان به دو صورت نمود دارد، یکی بصورت نمایانگرهای عددی که برای کارت و سخت‌افزار آن قابل فهم است و دیگری به صورت زبان اسمبلی که در اختیار و قابل فهم برای کاربر می‌باشد. بر روی کارت‌های Multos به هنگام مونتاژ و تولید، سیستم عامل Multos قرار میگیرد. این سیستم عامل بر روی چیپ‌های سیلیکون نوشته می‌شود.

۷-۱-۹- برنامه‌های کارت‌های Multos

در کامپیوترهای شخصی برنامه‌ها از طریق اجرای فایل‌های اجرایی آن بارگذاری می‌شوند. برنامه اجرا شده تا زمانی که از طرف کاربر بسته یا قطع نشود ، در حال اجرا می‌ماند. در دنیای کارت هوشمند اجرا شدن برنامه‌ها کمی متفاوت است. برنامه‌ها در کارت هوشمند در خلال اجرای دستوالعمل‌ها اجرا می‌شوند و پس از پایان دستورالعمل ، برنامه خاتمه می‌یابد.

۷-۱-۹-۱- انواع برنامه‌های کارت‌های Multos

برنامه‌های کارت‌های هوشمند به چند دسته تقسیم می‌گردد :

- برنامه‌های استاندارد
- برنامه‌های پوسته
- برنامه‌های واسط
- برنامه‌های Codelet

برنامه‌های استاندارد و پوسته در بخش‌های قبلی توضیح داده شده است. برنامه‌های واسط (Delegate) برنامه‌های استانداردی هستند که از طریق برنامه‌های دیگر و بصورت واسط و نماینده آن برنامه‌ها اجرا می شوند. Codeletها فایل‌هایی هستند که فقط شامل کدهای برنامه‌ها بوده و از طریق برنامه‌های دیگر فراخوانی می‌شود. ماهیت Codeletها همانند دیگر برنامه‌ها می‌باشد.

۷-۱-۱۰- Codelet چیست ؟

هنگامیکه برنامه‌های Multos را می نویسیم ، تصور ما بر اینست که تمامی اطلاعات برنامه در حافظه EEPROM در مراحل مختلف بارگذاری می‌شود. با جابجایی کدها و انتقال آنها به حافظه ROM امکان حفظ و ذخیره فضای EEPROM وجود دارد. این کدهای منتقل شده Codelet نامیده می‌شوند. Codeletها زمانیکه کارت در کارخانه یا کارگاه ساخته می‌شوند، بر روی کارت قرار میگیرند. یکی دیگر از ویژگی‌های Codeletها اینست که این برنامه‌ها می‌توانند از طرف هر برنامه‌ای فراخوانی و اجرا شوند.

۷-۱-۱۰-۱- ویژگی‌های Codeletها

- Codelet قسمتی از کد برنامه است که در حافظه ROM ذخیره شده است.
- Codelet می‌تواند از طریق هر برنامه‌ای در کارت فراخوانی گردد.
- هر Codelet دارای یک شاخص منحصربفرد است که توسط سیستم مدیریت احراز کلیدها تخصیص داده می شود.
- Codelet توسط زبان MEL نوشته شده و توسط AAM تفسیر می‌شود.
- Codelet از فضای برنامه فراخوانش برای فعالیت استفاده می‌کند. از اینرو امکان تسهیم اطلاعات بین برنامه‌های از طریق Codeletها وجود ندارد.
- Codeletها ممکن است خاص یک برنامه باشند.
- Codeletها ممکن است نگهدارنده کدهای عمومی مانند توابع کتابخانه‌ای باشند که در اختیار برنامه‌های مختلفی قرار می‌گیرند.
- Codeletها توسط برنامه‌های موجود در EEPROM فراخوانی گشته و پس از اتمام کار کنترل را به نقطه‌ای که Codelet فرخوانی گشته باز می گرداند.

- سیستم فراخوان Codelet ،۲ بایت از حافظه را جهت فراخوانی آن اشـغال میکنـد. اولی حاوی شاخص Codelet است که درباره آن توضیح داده شد و دومی آدرسـی است که Codelet در آن قرار داشته و باید به آن آدرس منحرف گردد.

۷-۱-۱۱- مراحل بارگذاری برنامه‌های Multos

مجموعه دستورات زیر برای بارگذاری و اجرای برنامه‌های در Multos استفاده می‌شود. در جدول(۷-۲) لیستی از دستورات و توضیح کوتاهی درباره آنها ذکر شده است.

جدول۷-۲ :دستورات بارگذاری برنامه‌ها در MULTOS

دستورات	ترتیب اجرا
باز کردن برنامه MEL	ابتدا
بارگذاری کدها و دستورات	مهم نیست
بارگذاری اطلاعات و داده‌ها	مهم نیست
بارگذاری پوشه فایل	مهم نیست
بارگذاری اطلاعات فایل	مهم نیست
بارگذاری Signature	مهم نیست
بارگذاری کلید واحد تبادل اطلاعات	مهم نیست
ایجاد برنامه MEL	انتها

دستوررات بارکردن برنامه MEL (Load MEL Application) اولین دستوراتی هستند که در فرایند اجرای برنامه فراخوانی می شوند. ترتیب فراخوانی دستورات بارگذاری (Load) مهم نبوده و در هر مرحله‌ای می‌توانند فراخوانی شوند. دستورات بارگذاری مجزایی برای هریک از بخش‌های کد، داده‌ها ، پوشه و اطلاعات فایل فراخوانی می‌شوند.

آخرین دستور در فرایند اجرا ایجاد برنامه (Create MEL Application) می‌باشد که برنامه‌ای از اطلاعات و فایل‌های بارگذاری شده ایجاد و فعال می‌کند. تنها در چند مورد فرایند بارگذاری متوقف می شود :

- کارت راه اندازی مجدد (Reset) شود
- برنامه دیگری در حال اجرا باشد
- دستور حذف برنامه به کارت ارسال گردد

اگر یک فرایند اجرا ناموفق بماند، تمامی اطلاعات آن از حافظه پاک شده و فضای حافظه در اختیار دیگر برنامه‌ها قرار می گیرد. در صورتیکه برنامه‌ای برای اجرا ناموفق بماند، Multos بنا را بر این می‌گذارد که یک تلاش برای دسترسی غیر مجاز صورت گرفته و بلافاصله شمارنده‌ای برای شمارش دفعات اجرای ناموفق فعال می‌کند. هنگامیکه شمارش معکوس این شمارنده به صفر برسد ، Multos تمامی دستورات وارد شده را برگشت داده و در حالت معلق و نیمه فعال قرار میگیرد.

۷-۲- بررسی سیستم عامل Java Card

در ژوئن ۲۰۰۵ یک بیلیون جاوا کارت فروخته شد؛ و این میزان بیشتر از سایر سیستم عامل‌های موجود کارت‌های هوشمند تا آن تاریخ است. این نتیجه منوط به شرایط اجتماعی‌اقتصادی، خوش شانسی و مهمتر از همه کیفیت فنی برتر جاوا کارت می‌باشد.

در اوایل دهه نود، سیستم عامل کارت هوشمندی با نام MASS مطرح شد؛ که بعدها Macsime نام گرفت و هم اکنون نیز با قابلیت‌ها و ویژگی‌های بسیار جاوا کارت نامیده می‌شود. دسته اول از این ویژگی‌ها در محصول [Che00] آمده، که به موفقیت‌های تجاری دست یافته است. دسته دوم از این ویژگی‌ها که در نمونه اولیه اجرایی Macsime عرضه شده، در محصول جاوا کارت معرفی شده است، محصولی برای گسترش هسته سیستم عامل کارت هوشمند که به سادگی قابل استفاده بوده و قابل حمایت در تجزیه و تحلیل‌ها است. مجموعه سوم و تکمیل کننده‌ای از ویژگی‌های جدید نیز برای جاوا کارت پیشنهاد شده که می‌تواند در برنامه‌های کاربردی خاص کاربرد داشته باشد.

محصولات جاوا کارت دارای ویژگی‌های نوین متعددی می‌باشند که قابلیت برنامه نویسی با زبان‌های سطح بالا (زیر مجموعه‌ای از زبان جاوا) برای برنامه‌های کاربردی کارت هوشمند، قابلیت ارتقا به برنامه‌های جدید، و مکانیزم قدرتمند امنیتی را دارا می‌باشند. این مکانیزمها بر پایه ایده‌های امنیتی جاوا و شی گرا بودن آن میباشد. بنابراین یک برنامه نویس جاوا کارت از متدهای طراحی شی گرا برای توسعه اپلتها و حتی کدهای ترمینال استفاده می‌کند؛ که به وسیله ابزارهای برنامه نویسی مانند emulatorها و debuggerها پشتیبانی می شود.

تکنولوژی جاوا کارت از دیدگاهی به نام روش مبتنی بر مدل استفاده می‌کند. این بدان معناست که نقطه شروع برای هر برنامه نویسی و توسعه‌ای، یک مدل انتزاعی از سیستم در حال

توسعه است. این مدل شامل تعاملات، احراز هویت، دسترسی و برقراری جایگاه تجاری سیستم است.

بسته به فضای گسترش و استقرار، ویژگی‌های جاوا کارت می‌توانند تغییر و افزایش یافته؛ ویژگی‌های متمم و تکمیلی مانند داشتن آدرس IP، قابلیت web-server بودن، ارتباط با دنیای خارج مثل صفحه کلیدها و صفحه نمایش ها.

۷-۲-۱- ویژگی‌های پیشنهاد شده برای جاوا کارت- توسعه مبتنی بر مدل‌های مختلف

یک کارت هوشمند بخشی از یک سیستم اطلاعاتی است که می‌توان آن را به این صورت معنی کرد: یک کارت هوشمند برای ترمینال در حکم slave و برای داده‌هایش در حکم master است. این نشان می‌دهد که یک کارت هوشمند ارتباط را پایه گذاری نمی‌کند، بلکه به درخواست‌ها پاسخ می‌دهد. کارت هوشمند در دسترسی به داده‌ها از روش‌های متفاوتی استفاده می‌کند و این روش‌ها به اپلت روی کارت بستگی دارد؛ ممکن است برای پاسخ به درخواست‌های ترمینال از برنامه‌ای به برنامه دیگر روش کنترل دسترسی به داده‌های ذخیره شده متفاوت باشد. به عبارت دیگر برنامه روی کارت تنها آگاهی لازم برای ارتباط با دنیای خارج را دارد.

۷-۲-۱-۱- مدل طراحی هم زمان کارت/ ترمینال

همانند دنیای سیستم‌های Embedded که در آن طراحی همزمان سخت‌افزار/ نرم‌افزار پذیرفته شده است؛ برای کارت‌های هوشمند نیز مدل طراحی همزمان کارت/ ترمینال پیشنهاد شده است که در آن برنامه‌های کاربردی کارت و ترمینال در یک محیط نوشته می‌شوند. سپس تیم‌های پشتیبانی اپلت کارت را از اپلت ترمینال مجزا می‌نمایند. اگرچه این مدل هنوز عملیاتی نشده است اما انتظار می‌رود که در آینده نزدیک، از آن در دنیای کارت‌های هوشمند استفاده شود.

امروزه بیشتر کارهای انجام شده سمت کارت بوده و متدها و ابزارهایی به کار گرفته شده اند که وظایف سمت کارت را به درستی اجرا کنند. هنوز توسعه ابزارهای سمت ترمینال جا برای کار دارند و توجه بیشتر تیم‌های تحقیقاتی را می‌طلبد.

۷-۲-۱-۲- مدل تعامل سه مرحله ای

بدون توجه به سایر بخش‌های طراحی برنامه‌های کاربردی، و با دقت در نحوه پاسخ کارت هوشمند به درخواست‌ها، متوجه می‌شویم که کارت سه عملیات را به طور یکسان در پاسخ به دستورات انجام می‌دهد؛ که در واقع مدل تعامل سه مرحله‌ای است:

فاز امنیت ورودی: کارت هوشمند به عنوان یک slave برای ترمینال، فرامین و داده‌ها را دریافت و رمزگشایی (decode) می‌کند. سپس در اکثر برنامه‌های کاربردی انتظار می‌رود که پردازش‌های امنیتی فعال شوند. به طور مثال پس از decode کردن داده‌های ورودی صحت امضا بررسی می‌شود.

فاز پردازش: هر داده‌ای که به کارت می‌رسد و یا در آن ذخیره می‌شود، می‌بایست پردازش شود.

فاز امنیت خروجی: پس از پردازش داده‌ها، یک فاز دیگر امنیتی وجود دارد که تضمین می‌کند داده‌های نتیجه قبل از ارسال به ترمینال به رمز می‌شوند.

هر برنامه نویس برنامه‌های کاربردی که از این مدل تعامل سه مرحله‌ای پیروی کند، متد مناسبی برای اجرای هر فاز مشخص می‌کند. برای کنترل هزینه‌های رمزنگاری، می‌توان از یک فاز ابتدایی برای تعیین تناسب درخواست و منابع موجود استفاده کرد.

۷-۲-۱-۳- مدل حافظه اجرایی Transacted

همانطور که در فاز پردازش اشاره شد، پردازش داده‌ها یک بخش ضروری در برنامه‌های کاربردی کارت است.پردازش برنامه‌های دردسترس، با مکانیزم ضد پارگی(anti-tear) که در آن کارت هوشمند قادر است در زمان قطع نا خواسته ارتباط با خواننده کارت، عکس‌العمل مناسب نشان دهد؛ متفاوت است. مدل حافظه اجرایی یک مکانیزم قابل استفاده در اپلت را ارائه می‌کند. این مدل می‌تواند جایگزین مکانیزم‌های anti-tear شده و تعداد به روز رسانی‌های وقت‌گیر حافظه غیر فرار را کاهش دهد.

به طور خلاصه می‌توان گفت مدل طراحی همزمان کارت/ ترمینال و مدل تعامل سه مرحله‌ای و مدل حافظه اجرایی، ایده‌های کلیدی کارت‌های هوشمند مدرن را شکل می‌دهند. جاوا کارت در حال حاضر این ویژگی‌ها را پشتیبانی نمی‌کند، اما هیچ مانع فنی برای افزودن این ویژگی‌ها به تکنولوژی جاوا کارت وجود ندارد.

۷-۲-۲- ویژگی‌های موجود جاوا کارت- برنامه نویسی شی گرا

اپلت‌های جاوا کارت برنامه‌های کوچکی هستند که می‌توانند با یکدیگر و یا با ترمینال ارتباط برقرار کنند؛ با زبان‌های سطح بالا نوشته می‌شوند و سپس روی کارت هوشمند سوار می‌شوند. سایر سیستم عامل‌های کارت‌های هوشمند برنامه نویسی با زبان‌های سطح بالا(مانند MULTOS) و یا ارتباط بین اپلت‌ها را میسر می‌سازند. نوآوری تکنولوژی جاوا کارت در این است که مجموع این ویژگی‌ها را که در واقع نشان از شی گرا بودن آن است؛ پشتیبانی می‌کند.

۷-۲-۲-۱- بار گذاری اپلت

بار گذاری یک اپلت به معنی بار گذاری یک کلاس است. امنیت اپلت‌های بار گذاری شده به طور مستقیم با کنترل دسترسی به کلاس‌ها و شی نتیجه در ارتباط است. این ایده مستقیماً از خود جاوا آمده است، که در آن load ایمن کلاس‌ها ارائه شده است.

۷-۲-۲-۲- ارتباط بین اپلت‌ها

اپلت‌ها به وسیله share کردن اشیا با یکدیگر ارتباط برقرار می‌نمایند. همه شی‌ها نمی‌توانند به وسیله همه اپلت‌ها share شوند، به همین علت heap جاوا به تعدادی از heapهای مجازی تقسیم می‌شود. این نوع از کنترل دسترسی به عنوان firewall جاوا کارت شناخته شده است که نام گذاری غلطی می‌باشد.

۷-۲-۲-۳- تعیین صحت بایت کد

به عنوان یک بخش از load شدن اپلت، محیط اجرایی جاوا کارت، صحت بایت کد تولید شده را بررسی می‌کند. این کار در دو مرحله انجام می‌شود؛ یکی هنگامی که اپلت در وضعیت off-card است و دیگری هنگامی که اپلت روی کارت نصب شده و در حالت on-card می‌باشد. در هر دو حالت فرضیه حتمی این است که اپلت پس از یک بار تعیین صلاحیت شدن، نمی‌تواند تغییر کند. این فرضیه ممکن است بسیار قدرتمند باشد، مخصوصاً اگر کدهای محلی بتوانند روی کارت اجرا شوند.

۷-۲-۲-۴- مروری بر ویژگی‌ها

جدول(۷-۳) گذاری بر ویژگی‌های سیستم عامل‌های کارت هوشند را نشان می‌دهد و آن را با ویژگی‌هایی که در نسخه جدید جاوا کارت عرضه شده است مقایسه می‌کند.

جاوا کارت و MULTOS سیستم عامل‌هایی هستند که در قسمت‌های قبل و همچنین اینجا به آنها پرداخته شده است. اما دیگر سیستم عامل‌هایی که در این جدول آمده است جزء سیستم عامل‌های متن باز نبوده و به همین خاطر از پرداختن به آنها صرفنظر شده است.

ویژگی‌هایی که در این جدول آمده است به شرح زیر می‌باشد:

Multi Application – چند برنامه‌ای: این بدین معنی است که نقش کارت می‌تواند برای پشتیبانی عملیات مختلف برای یک هدف خاص، انتخاب شود. تقسیم کردن کارت به بخش‌های اجرایی مجزا یک مکانیزم ضروری برای کارت‌های چند برنامه‌ای است.

Interpreter - مفسر: این بدین معنی است که سیستم عامل کارت دارای یک ماشین مفسر مجازی است که کدهای برنامه را تفسیر می‌کند.

Application Programming Language – زبان برنامه نویسی برنامه کاربردی: زبانی که برنامه کاربردی با آن نوشته و اجرا می‌شود.

Dynamic – دینامیک: ویژگی ای که در آن کارت‌های چند برنامه‌ای می‌توانند پس از تولید، به وسیله حذف یا اضافه کردن برنامه‌های کاربردی که از یک مکانیزم مدیریت کارت استفاده می‌کنند، تغییر کند.

Card Management – مدیریت کارت: مکانیزم ویژه‌ای که برای مدیریت کارت استفاده می‌شود و بر روی کارت‌ها با سیستم عامل‌های خاص اجرا می‌شوند.

Cross Firewall Communication: یک مکانیزم که به برنامه‌های کاربردی اجازه می‌دهد به صورت ایمن در درون کارت با یکدیگر ارتباط داشته باشند.

Program-accessible transaction model: یک مدل از تعاملات که به طور خاص مربوط به کارت‌های هوشمند است.

3-phase Interaction Model: یک مدل سه مرحله‌ای از تعاملات بین کارت هوشمند و محیط اطراف آن.

به طور خلاصه، ادغام جاوا و کارت هوشمند یک محیط قدرتمند برای توسعه برنامه‌های کاربردی کارت هوشمند پدید آورده است.

۷-۲-۳- ویژگی‌های تکمیلی

در این قسمت به برخی از ویژگی‌های تکمیلی جاوا کارت که تا کنون ارائه شده است می‌پردازیم.

۷-۲-۳-۱-آدرس دهی به کارت با یک شماره منحصر به فرد

یکی از ویژگی‌های تکمیلی که به صورت گسترده برای جاوا کارت پیشنهاد شده است؛ تهیه یک آدرس IP برای کارت هوشمند با استفاده از پروتکل ارتباطی TCP/IP است. جاوا کارت بدون آدرس IP همانند یک سرور بی نام کار می‌کند. این می‌تواند به امنیت و حفاظت از اطلاعات کمک کند. بی نام بودن کارت یک ویژگی ضروری برای کارت‌های چند برنامه‌ای است؛ و برای جاوا کارت این بی نامی امکان کار اپلت با برنامه‌های کاربردی را بدون نگرانی از حمله‌های امنیتی میسر می‌سازد.

جدول۷-۳: ویژگی‌های سیستم عامل‌های کارت هوشمند

Technology / Feature	Oscar	Macsime/ TOSCA	Blue	Basic Card	MULTOS	Java Card
Source	GIS	QC technology/ Integrity Arts	Digi-Cash	Zeit Control	NWDT	Sun Microsystems
Country	UK	NL/ US	NL	GE	UK	US
Introduction	1990	1994-1995	1996	1997	1996	1997
Multi-application/ firewall	X	X			X	X
Interpreter		SCIL/ CLASP	J-Code	ZC byte code	MEL	Java Card Byte Code
Application programming language		Data model/ ADA	J-Code	Basic	MEL/ C	Java
Dynamic	X	X			X	X
Card Management	Master file			One-shot loader	Multos	(VISA) Open Platform
Cross-firewall communication		X			Via I/O buffer	API
Program accessible Transactions		X				Single
3-phase interaction model		X				

در برخی از برنامه‌های کاربردی؛ آسیب کمی در آدرس دهی به کارت هوشمند از طریق یک شناسه واحد وجود دارد. برای مثال سیم کارت‌های تلفن که یکی از رایج ترین برنامه‌های کاربردی جاوا کارت است؛ انباشته از شناسه‌های مختلف (شماره تلفن، PIN Code و ...) است، که می‌تواند آسیب‌پذیر باشد.

در برنامه‌های دیگر، آدرس دهی کارت هوشمند با یک شناسه منحصر به فرد مضر است. برای مثال اینکه کارت‌های بانک اطلاعات واضحی از پرداخت‌های به مغازه‌ها، هتل‌ها، رستوران‌ها و ... با خود به همراه داشته باشند, مورد علاقه نیست. یک کارت بانک، شماره حساب را در خود دارد و این شماره حساب به وسیله برنامه درون کارت محافظت می شود و به آسانی از دنیای خارج کارت در دسترس نمی باشد. یک کارت نباید شناسه‌های منحصر به فرد خود، همانند آدرس IP را اعلان کند. چنین شناسه‌ای ID نام دارد.

۷-۲-۳-۲- تعامل بیشتر با محیط

مدل اولیه پردازشگر جاوا کارت، جدا از متدهای ارتباطی بین کارت و ترمینال است و از پروتکل انتقال بسته‌ای اطلاعات که یک استاندارد بین المللی است؛ و پروتکل پاسخ دهی به فرامین در لایه برنامه کاربردی، پیروی می‌کند. هنگام تهیه جزئیات ارتباطی، ویژگی RPC، بسیاری از محدودیت‌های زبان جاوای جاوا کارت را برای طراحان برنامه‌های کاربردی ترمینال روشن کرد.

۷-۲-۳-۳-آزاد کردن کارت از بردگی

برخی از دانشمندان اعتقاد دارند که کارت باید از حالت slave خارج شده و نقش فعال تری را دارا باشد. دلیل آنها برای این پیشنهاد این است که یک کارت فعال می‌تواند سرویس دهی به برنامه‌های کاربردی را به طور همزمان انجام دهد و taskهای متعددی را اجرا نماید. اگرچه می‌دانیم که این ویژگی امنیت کارت را تا حد زیادی تضعیف می‌نماید. چند برنامه‌ای بودن شانس مواجهه با اجرای برنامه‌هایی را که می‌توانند به عنوان جاسوس عمل کنند را بالا می‌برد. بنابر این به تحقیقات بیشتری در این زمینه نیاز است تا بتوان تاثیر فعال تر کردن کارت را در بحث امنیت آن بررسی کرد.

۷-۲-۳-۴- افزودن APIهای جدید

جاوا کارت از تعدادی از APIها برای رمزنگاری و سایر عملیات استفاده می‌کند. ولی هنوز بسیاری از APIهای مورد نیاز توسط جاوا کارت پشتیبانی نمی‌شود. شاید این امر نیز یکی از مباحث مورد بحث جاوا کارت در دنیای امروز باشد.

مراحل ساخت یک برنامه کاربردی برای جاواکارت به شرح زیر است:

- نوشتن کدهای جاوا
- کامپایل کردن کدهای نوشته شده
- تبدیل فایل کامپایل شده به فایل CAP (Converted Applet File)
- بررسی صحت فایل CAP (انجام این بخش اختیاری است)
- نصب فایل CAP

دو مرحله اول با مراحل برنامه‌نویسی رایج در زبان جاوا یکسان می‌باشد. نوشتن فایل‌های java و کامپایل آنها به فایل‌های class برای جاواکارت متفاوت می‌باشد.

قبل از اینکه کلاس‌های جاواکارت بتوانند در جاواکارت load شوند، باید به قالب یک فایل CAP استاندارد تبدیل شده و سپس در صورت تمایل صحت فایل بررسی شود. پس از آن فایل CAP برای نصب شدن بر روی جاواکارت آماده است.

۷-۳- برنامه نویسی جاوا کارت

برنامه نویسی جاوا کارت دنیای جدیدی را برای برنامه‌های کاربردی کارت هوشمند به ارمغان آورده است. کارت‌جاوا دارای یک ماشین مجازی جاوا(Java Virtual Machine – JVM) می‌باشد. برنامه‌های جاوا می‌توانند بر روی کارت ذخیره و اجرا شوند. برنامه نویسی جاوا کارت مبتنی بر ویژگی‌های آن است که توسط شرکت Sun ارائه می‌شود. ویژگی‌های JVM موجود بر جاوا کارت به صورت زیر است:

یک نسخه محدود از JVM که زیرمجموعه‌ای از زبان برنامه نویسی جاوا را که به وسیله اپلت‌های جاوا کارت استفاده می‌شوند پشتیبانی می‌کند.

یک API برای توسعه اپلت‌های کارت هوشمند که بر پایه استانداردهای ISO 7816 در دسترس قرار دارد و توسعه برنامه‌های کاربردی را پشتیبانی می‌کند.

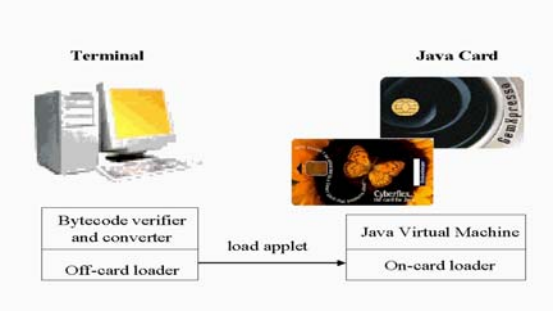
یک محیط زمان اجرای کوچک که مدیریت اپلت‌ها را بر عهده دارد؛ همانند مکانیزم انتخاب اپلت ها. این محیط JCRE(Java card Runtime Environment) نامیده می‌شود.

با توجه به محدودیت‌های فنی موجود در پردازنده کارت‌های هوشمند و از آنجائیکه برخی از ویژگی‌ها مانند multithreading جز ضروریات جاوا کارت نمی‌باشد؛ تنها زیرمجموعه‌ای از زبان جاوا در کارت‌های جاوا پشتیبانی می‌شود. اجرای یک JVM، از یک bytecode verifier، یک class loader و یک bytecode interpreter تشکیل شده است. Verifier برای تشخیص اینکه کلاس مورد استفاده جزء کلاس‌های معتبر جاوا باشد، استفاده می‌گردد. Loader برای بارکردن کلاس‌ها در سیستم استفاده می‌شود. Interpreter نیز برای اجرای برنامه‌های کاربردی مورد استفاده قرار می‌گیرد.

Bytecode verifier یک نرم‌افزار بزرگ و پیچیده است که نمی‌تواند بر روی کارت هوشمند سوار شود. بنابراین اجرای JVM در کارت هوشمند به دو بخش زیر تقسیم می‌شود.

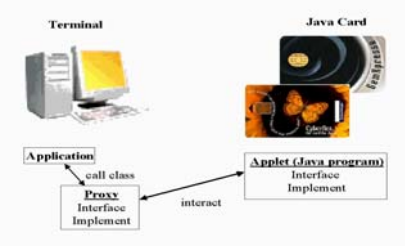
بخش off-card که بررسی کلاس‌ها را مدیریت کرده و این اطمینان را به ما می‌دهد که تمامی کلاس‌های لازم در دسترس قرار دارند. بخش on-card نیز اساساً برای اجرای بایت کدها کاربرد دارد.

JVM یک ماشین مستمر و پایدار است؛ در نتیجه موقعیت برنامه‌ها و objectها حتی پس از قطع برق از کارت نیز نگهداری می‌شود. داده‌های مرتبط نیز در EEPROM ذخیره می‌گردند. کار دیگر JVM این است که کلاس‌ها را بار کرده و یک بار مقدار دهی اولیه می‌کند؛ در نتیجه این کلاس‌ها تا پایان کار فعال می‌مانند.



شکل۷-۶: معماری Bytecode Verifier در جاوا کارت

در کنار متدولوژی استاندارد دستور/ پاسخ APDU، یک راه دیگر استاندارد برای تعامل با یک برنامه در جاوا کارت، استفاده از متدولوژی فراخوانی از راه دور (RMI) می‌باشد. RMI یک تکنولوژی شی گرای توزیع شده است و نوعی از معماری است که در آن هر سرویسی که به وسیله یک سرور (کارت هوشمند) ، ارائه می‌شود؛ می‌بایست دارای یک لایه واسط(interface) باشد. این لایه واسط لیستی از متدها را که برای یک object خاص در دسترس هستند،مشخص می‌کند.



شکل۷-۷ : proxy بین برنامه کاربردی و اپلت

یک لایه واسط با این ویژگی‌ها همانند قراردادی است که سرور را به سرویس گیرنده (ترمینال)، مقید می‌نماید. سرور تضمین می‌کند که به کلیه متدهایی که در لایه واسط خود دارد پاسخ دهد. به عبارت دیگر این پروتکل سرور را به سرویس گیرنده متصل می‌کند و طریقه ارتباط آنها را تشریح می‌نماید. از آنجایی که اجرای این پروتکل‌ها عموماً پیچیده است، اجرای آنها به طور اتوماتیک در شی JCRE تولید می‌شود. این برنامه اتوماتیک تولید شده، که پروتکل سمت سرویس گیرنده را اجرا می‌کند؛ معمولاً Proxy نامیده می‌شود.

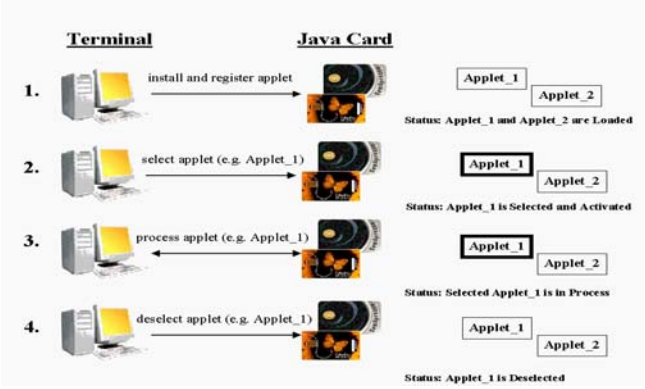
علاوه بر داشتن کدهای مربوط به توابع، proxy کدهای مربوط به دسترسی به این توابع از طریق یک سرور راه دور را نیز در خود دارد. جاوا کارت می‌تواند به عنوان یک سرور در نظر گرفته شود که سرویس‌هایی را برای دسترسی و مدیریت داده‌های ذخیره شده در کارت هوشمند به سرویس گیرندگان خود (ترمینال‌ها) ارائه می‌کند. علاوه بر این پروتکل‌های مختلفی که در استاندارد ISO 7816-3 و 4 وجود دارد، کارت هوشمند را به عنوان برده در رابطه رئیس/ برده توصیف می‌کنند. قدرت اجرایی برنامه جاوای موجود در جاوا کارت (اپلت)، در لایه واسط نمایان می‌شود؛ لایه‌ای که در آن لیستی از متدهای در دسترس تشریح شده است. یک پروتکل سطح بالا ارتباط بین اپلت و سرویس گیرنده (ترمینال) را مشخص می‌کند. یک مولد proxy برای طراحی و توسعه نرم‌افزار سرویس گیرنده در دسترس می‌باشد.

سه قانون اصلی برای کنترل امنیت و قابلیت مشاهده و دسترسی اپلت‌ها روی جاوا کارت وجود دارد:

- قابل مشاهده و در دسترس بودن یک بسته (package) وابسته به پلتفرم است.
- در یک بسته قابل مشاهده، فقط کلاس‌های عمومی از بیرون قابل مشاهده و در دسترس هستند.
- اگر یک اپلت می‌تواند به مرجع یک شی دسترسی داشته باشد؛ آنگاه اجازه دارد که از آن شی استفاده کند.

در واقع این سه قانون با قوانین استاندارد جاوا یکی است. علاوه بر این اکثر سازندگان جاوا کارت، ویژگی‌های امنیتی (firewall) بیشتری را بین اپلت‌ها قرار می‌دهند. این ویژگی‌ها در کارت، عمومی هستند و برای در قرنطینه نگاه داشتن هر شی استفاده می‌شوند تا از دسترسی غیرقانونی به آن جلوگیری شود.

پس از اینکه یک اپلت جاوا کارت ایجاد شد و بر روی ترمینال سوار شد، اولین مرحله نصب و ثبت آن به جاوا کارت می‌باشد. زمانیکه اپلت با موفقیت ثبت شد، آماده است تا انتخاب شده و فعال گردد.



شکل۷-۸: چرخه حیات یک اپلت در جاوا کارت

در هر لحظه فقط یک اپلت می‌تواند انتخاب شده و فعال گردد. چناچه انتخاب اپلت با موفقیت صورت گیرد، اپلت برای پردازش دستورات آماده است. در زمانی که یک اپلت انتخاب شده است، هر دستوری که به کارت فرستاده می‌شود، در داخل یک APDU قرار دارد و به متد

پردازش اپلت ارسال می‌گردد. این روند تا زمانی که اپلت از حالت انتخاب خارج شود اداده پیدا می‌کند. متد غیرفعال کردن هر اپلت باید پیش از فعال شدن اپلت دیگر اجرا شود

۷-۴- انتخاب سیستم عامل متن باز مورد نظر

پلتفرم جاوا کارت یک محصول تجاری موفق است، زیرا نیازهای اپراتورهای تلفن همراه را برای ارتباط با کارت‌های هوشمند بر آورده می سازد. تکنولوژی جاوا کارت تعداد قابل توجهی از تحقیقات در زمینه سیستم های عامل، مهندسی نرم‌افزار و زبان‌های برنامه نویسی را به خود اختصاص داده است. محققان این پلتفرم را محیط جالبی برای پیاده سازی ایده‌های نوین و یا برای تست ایده‌های قدیمی که در جاوا کارت مطرح شده است؛ یافته اند.

سیستم جاوا کارت دارای ویژگی‌های قدرتمندی است که برخی از آنها تست و اثبات شده‌اند (مانند شی گرا بودن) و برخی دیگر نیز بسیار نوین و خلاقانه هستند (مانند اپلت firewall).

محیط جاوا کارت به عنوان یک ابزار برنامه نویسی و توسعه هنوز در آغاز راه است. مدیریت چرخه زندگی کارت و توسعه برنامه‌های کارت/ ترمینال هنوز در مراحل آزمایشگاهی به سر می‌برند.

تلفیق و همسانی دگرگونی‌های امنیتی در برنامه‌های کاربردی در یک سیستم عامل با روال‌هایی با امنیت بالا در صورتی امکان پذیر است که شرایط و محدودیت‌های زیر دست یافتنی باشد :

- تمام منابع الحاق شده به یک برنامه بایستی بطور کامل مشخص شده و دارای شاخص منحصر بفرد باشد. این منابع شامل کدهای برنامه ، داده‌های برنامه ، کلیدهای شاخص‌های برنامه‌ها و دیگر ملزومات برنامه‌ها باشند که ممکن است خارج از فضای خاص سیستم عامل قرار گیرند و یا ارجاعی به منابع مجاز در برنامه‌های دیگر باشند.
- مدیریت این منابع در طول دوره حیاط و اجرای برنامه از ایجاد و آغاز تا خاتمه می‌بایست مشخص شده و تعریف شده باشد.
- سیستم عامل بایستی دارای مکانیزم دیواره امنیتی یا دیواره آتش جهت مجزا کردن منابع برنامه‌ها باشد.

- اشتراک و تسهیم منابع میان برنامه‌ها می‌بایست طبق تعریف دسترسی‌های مجاز و بصورت کاملا شفاف و معتبر انجام پذیرد.

۷-۴-۱- انتخاب پلتفرم متن باز

با گسترش روزافزون استفاده از جاوا کارت ، MULTOS و کارت‌های ویندوز برای کارت‌های هوشمند، واژه پلتفرم متن‌باز رواج پیدا کرده است. این اصطلاح به معنی متن‌باز بودن سیستم‌عامل کارت هوشمند می‌باشد که اجازه می‌دهد افراد مختلف برنامه‌های مختلف به کارت هوشمند اضافه نمایند ، بدون آنکه جزئی از سازندگان آن سیستم عامل باشند.

نقطه مقابل پلتفرم های متن‌باز، پلتفرم‌های اختصاصی هستند که متعلق به یک شرکت خاص می‌باشند. در واقع هم اکنون در دنیای کارت‌های هوشمند، واژه پلتفرم متن‌باز معنای واقعی و صحیح خود را نیافته است ؛ زیرا همانند Linux دسترسی مجانی و آزادانه به کدهای آنها، عدم وجود محدودیت‌های licensing و عدم وابستگی به شرکت‌ها یا موسسات خاص در مورد آنان صدق نمی‌کند.

در حال حاظر هیچ تولید کننده کارت هوشمندی تمامی مشخصات یا Specification محصول خود را به طور کامل در اختیار برنامه سازان متن باز قرار نمی‌دهد و معمولاً تمامی امکانات اصلی و برنامه‌های در نظر گرفته شده برای کارت را تولید کنندگان خودشان بر روی کارت‌ها قرار داده و تنها امکان اضافه یا کم کردن بخش‌هایی از برنامه‌ها را تحت شرایط تعریف شده‌ای به دیگران می‌دهند.

۷-۴-۲- Multos انتخاب بهتر

همانطور که در قسمت های قبل گفته شد؛ Multos یک سیستم عامل چند برنامه برای کارت‌های هوشمند می‌باشد و از توسعه Mondex به دست آمده است. Multos سیستم عامل بهینه‌ای برای برطرف کردن نیازهای سیستم‌های پرداخت الکترونیکی است که توسط کنسرسیوم Maosco حمایت می‌شود. یکی از جالبترین و مهم‌ترین ویژگی‌های Multos این است که از استاندارد ITSEC E6 پیروی می‌کند، که در بالاترین سطح امنیتی قرار دارد. این بدان معنی است که Multos جز امن‌ترین سیستم عامل‌های موجود برای کارت‌های هوشمند است.

Multos برای کامپایل و اجرای برنامه‌ها از زبان مخصوص به خودی به نام MEL استفاده می‌کند. MEL مستقل از سخت‌افزار می‌باشد و توسط یک ماشین مجازی به نام AAM اجرا می‌شود.

با توجه به توضیحاتی که در فازهای قبل در مورد ویژگی‌های جاواکارت و Multos داده شد؛ انتخاب بین جاواکارت و Multos که هر دو جز قابل‌ترین و مطرح‌ترین سیستم عامل‌های کارت هوشمند در دنیا می‌باشند ، انتخاب بسیار سخت و زمان بری می‌باشد. هر دوی این سیستم عامل‌ها از قابلیت‌های سطح بالایی برخوردار بوده که انتخاب را دشوار می‌سازد. اما با توجه به نیازهای بومی و منطقه‌ای که در کشور وجود دارد، به نظر می‌رسد که Multos بتواند گزینه بهتری باشد. زیرا هردوی این سیستم عامل‌ها چند برنامگی را پشتیبانی کرده و با زبان‌های سطح بالای برنامه نویسی کار می‌کنند. اما Multos از لحاظ امنیتی و برنامه نویسی قابلیت‌های بیشتری نسبت به جاواکارت دارد. همواره برای انتخاب یک محصول بعنوان محصول ملی و جهت بسط آن بعنوان زیر ساخت، می‌بایست مجموعه کاملی از عوامل را در نظر گرفت. در کشور ما استفاده از کارت هوشمند در سطح ملی مستلزم تعریف نیازها و بررسی ساختارهای استفاده کنندگان و سرویس دهندگان می‌باشد. بسیاری از سازمان‌ها و ارگان‌های ملی، سرویس‌های گوناگونی را در اختیار عموم قرار می دهند و انتظار دارند که بتوانند تمامی آن سرویس‌ها را در قالب یک کارت هوشمند به طرفین قرار داد خود ارائه دهند. همچنین عدم وجود بسترهایی که کارایی خود را در برقراری امنیت انتقال اطلاعات ثابت کرده باشند، سرویس دهندگان را ملزم به اعمال امنیت انتقال اطلاعات در پایانه‌های ارائه خدمات و همچنین در ابزارهای در اختیار مردم، می کند. بدان معنی که وجود مکانیزم پیشرفته‌ای در ترمینال‌ها و پایانه‌های کارت های هوشمند و همچمنین خود کارت‌ها الزامی و مورد نیاز می‌باشد. همچنین هزینه‌های خطوط ارتباطی و تاخیر زمانی در انتقال اطلاعات در این خطوط می‌بایست از طرف ترمینال‌های کارتخوان و همچنین خود کارت‌ها جبران گردد. بدین معنی که وجود کارت‌هایی که عملیات را در کمترین زمان ممکن انجام داده و به ترمنیال انتقال دهند؛ یا بعبارتی زمان دسترسی به اطلاعات در آنها کم باشد؛ جبران کننده بقیه تاخیرهای زمانی در بسترهای ارتباطی می باشد.

Multos بعنوان محصول انتخابی این مستند تمامی ویژگی‌های یاد شده را داشته و در مقایسه با دیگر محصولات موجود، بهترین انتخاب می باشد. جاوا کارت بعنوان یکی دیگر از

محصولات مطرح شده در این بحث، دارای تکنولوژی نوین و نوپایی بوده و بسیاری از ویژگی‌ها و توانایی‌های Multos، در حال حاظر در جاواکارت در مراحل شکل گیری و تست قرار دارد و نیازمند گذشت زمان برای عملیاتی شدن بعنوان بستر ملی می باشد. در حال حاظر بسیار از محصولات مبتنی بر کارت‌های هوشمند در دنیا بر پایه و اساس Multos شکل گرفته‌اند و امنیت Multos بحدی بالا بوده که MasterCard بعنوان معتبرترین ابزار پرداخت الکترونیکی در دنیا بر اساس آن ایجاد و راه اندازی گشته است. علاوه بر امنیت در نقل و انتقالات مالی، وجود قابلیت‌های چند برنامگی امکان وجود چنیدین سیستم و برنامه را در کارت‌های Multos چنان فراهم آورده که حتی می‌توان برنامه‌هایی را که کمترین وابستگی اطلاعاتی و مفهومی را بهم دارند در نهایت امنیت و تجرد ، در کنار هم بر روی کارت قرار داد. مانند سیستم های احراز هویت ، سوابق ، اطلاعات درمانی ، بیمه ای ، پرسنلی و .... در زیر برای پایان بخشیدن به مبحث انتخاب سیستم عامل متن باز مناسب برای کشور ، مجموعه‌ای از ویژگی‌های Multos را در قیاس با دیگر رقبا بررسی می‌نماییم.

جدول ۷-۴: زمانبندی تراکنش‌های موجود کارت‌های هوشمند

Platform: / Transaction description	Native C (Softmask) (SLE66CX160S)	Native C (Bondout) (SLE44CR80S)	MULTOS V4 (MEL) (SLE66CX160S)	MULTOS v3 (MEL)	MULTOS v4 C SLE66CX160S	MULTOS v4 C HSMA410b	SmartCafé (SLE66CX160S)
Off-line transaction accepted - Crypto enabled - Cumulative Amount test executed	3.74		6.06	6.5 s	6.75 s	6.1 s	8.6 s
Off-line transaction accepted - Crypto disabled - Cumulative Amount test executed	3.61		5.8	6.3 s	6.51 s	5.77 s	7.3 s
Off-line transaction accepted - Crypto enabled - Cumulative Amount test not executed	3.66	3.9 s	5.73	5.8 s	6.39 s	5.9 s	8.3s
On-line transaction accepted - Crypto enabled - Cumulative Amount test executed - Issuer authentication	4.2		6.72	7.4 s	7.6 s	7.0 s	11.2 s

جدول۷- ۵: زمانبندی دستورات اجرایی در کارت‌های هوشمند

Command / Platform:	Native C Softmask SLE66CX160S	Native C bondout SLE44CR80S	MULTOS v4 MEL SLE66CX160S	MULTOS v3 MEL	MULTOS v4 C SLE66CX160S	SmartCafé Java Card SLE66ex160S
Selection of an application that is present	150 ms	150 ms	300 ms	170 ms	320 ms	300 ms
Get Processing Options	140 ms	140 ms	310 ms	440 ms	330 ms	300 ms
Read record (size 57 bytes)	185 ms	175 ms	230 ms	300 ms	250 ms	170 ms
Read record (size 116 bytes)	350 ms	325 ms	390 ms	360 ms	390 ms	310 ms
Get Data (PTC)	50 ms	50 ms	70 ms	60 ms	130 ms	110 ms
Verify PIN	110 ms	120 ms	120 ms	260 ms	190 ms	220 ms
Generation of AAC required by terminal	360 ms	460 ms	610 ms	800 ms	750 ms	1900 ms
Generation of ARQC required by terminal	380 ms	480 ms	920 ms	1200 ms	1200 ms	3000 ms
Generation of TC - cumulative amount check executed	400ms	500ms	1290ms	1430ms	1650ms	3210 ms
Generation of TC - cumulative amount check not executed	380 ms	500 ms	1020 ms	1170 ms	1170 ms	2860 ms
Generation of TC at 2nd generate AC without Issuer authentication	400 ms	480 ms	770 ms	910 ms	1070 ms	2520ms
Generation of AAC at 2nd generate AC with Issuer authentication failed	360 ms	470 ms	580 ms	760 ms	940 ms	2430 ms

جدول۷-۶: زمانبندی اجرای برنامه های رمزنگاری شده

Platform: / Command	Native C Softmask SLE66CX160S	Native C boudout SLE44CR80S	MULTOS v4 MEL SLE66CX160S	MULTOS v3 MEL	MULTOS v4 C SLE66CX160S	Sm@rtCafé SLE66CX160S
Generation of the application cryptogram (only crypto: key diversification and MAC)	130 ms	<200 ms	300 ms	200 -> 350 ms		900->1300 ms
Cumulative amount check (only token execution)	20 ms		270 ms	260 ms	480 ms	350ms

جدول۳-۷: مقایسه تولید برنامه‌های اعتباری در کارت های هوشمند

Aspect / Platform	Native C	MULTOS SwiftCard	MULTOS MDS	Sm@rtCafé	Windows for Smart Cards
Development Environment	++	+	-	+++	+
Code size	++	- for C ++ for MEL	++	-	---
Timing performance	+++	++ for MEL - for C	++	--	?
Validation using Europay CTA Tool	+	+	+	+	-

فصل هشتم – بررسی و شناسایی تغییرات لازم جهت بومی سازی سیستم عامل کارت هوشمند

۸–۱– عمومی سازی و بومی سازی

عمومی‌سازی (Internationalization) به فرآیند طراحی یک نرم‌افزار به‌منظور تطبیق با زبان‌ها و کشورهای مختلف بدون نیاز به تغییر در کد برنامه اطلاق می‌شود. معمولاً بیشترین زمان فاز بومی‌سازی صرف ترجمه متن می‌شود. سایر اطلاعات از قبیل صدا و تصویر در صورت نیاز بومی‌سازی می‌شود. تیم بومی‌ساز نرم‌افزار همچنین امکان نمایش اطلاعاتی از قبیل تاریخ، اعداد و پول را نیز برای یک منطقه خاص فراهم می آورند. یک برنامه عمومی شده قابلیت پشتیبانی زبان‌های مختلف و پارامترهای وابسته مانند تاریخ، زمان، پول و... را بدون نیاز به تغییر کد برنامه دارا می‌باشد. این ایده تحت عنوان (soft coding) یا جداسازی اجزاء رابط کاربر از کد منطق برنامه نیز مطرح می‌شود.

بومی‌سازی (Localization) به فرآیند تطبیق نرم‌افزار برای یک منطقه یا زبان خاص با اضافه کردن اجزا مربوط به آن منطقه و نیز ترجمه پیغام‌ها اطلاق می‌شود. معمولاً بیشترین زمان فاز بومی‌سازی صرف ترجمه متن می‌شود . سایر اطلاعات از قبیل صدا و تصویر در صورت نیاز بومی‌سازی می‌شود. تیم بومی‌ساز نرم‌افزار همچنین امکان نمایش اطلاعاتی از قبیل تاریخ، اعداد و پول را برای یک منطقه خاص فراهم می‌آورند.

لازم به ذکر است که در مقالات و نوشته‌های فنی، به‌جای کلمه Internationalization از مخفف آن I18N استفاده می‌شود. توضیح اینکه بین کاراکترهای شروع و پایان این کلمه ۱۸ کاراکتر وجود دارد. بنابراین برای سهولت از کلمه I18N و همچنین از I18N'd به جای Internationalized استفاده می‌شود.

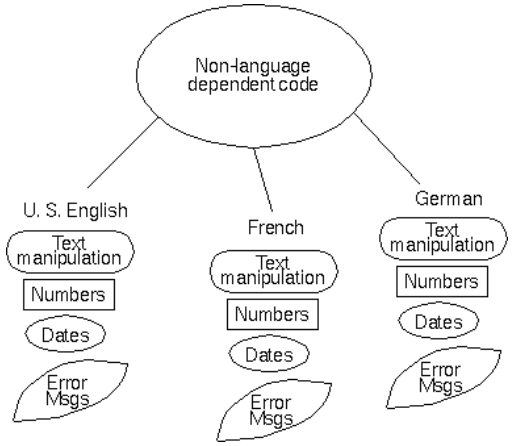
۸–۱–۱– عمومی سازی Internationalization

یک برنامه عمومی شده دارای مشخصات زیر است:

- در کنار یک‌سری اطلاعات بومی شده، برنامه می‌تواند در هر کشور و به هر زبانی اجرا شود.

- عناصر متنی مانند پیغام‌ها و عنوان اجزا رابط کاربر، در داخل کد برنامه (hard coding) مشخص نمی‌شوند، بلکه در یک منبع خارج از کد برنامه ذخیره و بصورت دینامیک بازیابی می‌شوند.
- پشتیبانی از زبان‌های جدید بدون نیاز به کامپایل مجدد برنامه.
- اطلاعات حساس به منطقه جغرافیایی از قبیل تاریخ، پول و اعداد مطابق با استانداردهای تعریف شده در کشور کاربر نهایی نمایش داده می‌شود.
- به آسانی قابل بومی‌سازی است.

در شکل (۸-۱) مثال‌هایی از ساختار کدهای غیروابسته به زبان نشان داده شده است؛ که در آن عناصر متنی مانند پیغام‌ها و عنوان اجزا رابط کاربر، در داخل کد برنامه (hard coding) مشخص نمی‌شوند، بلکه در یک منبع خارج از کد برنامه ذخیره و بصورت دینامیک بازیابی می‌شوند.



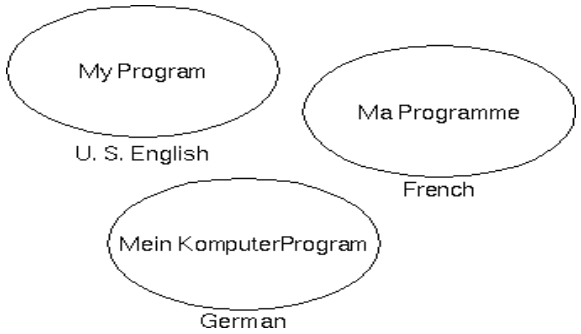
شکل ۸-۱: ساختار کدهای غیروابسته به زبان

۸-۱-۲- بومی سازی Localization

بومی‌سازی به فرآیند طراحی و پیاده‌سازی یک نرم‌افزار با قابلیت تطابق با یک زبان، فرهنگ، کشور و منطقه خاص گفته می‌شود. بسیاری از برنامه‌های نوشته شده برای یک کشور یا منطقه خاصی تنها در آن محدوده جغرافیائی قابل استفاده می‌باشند.

به‌جای کلمه Localization از مخفف آن L10N استفاده می‌شود. این مخفف نیز همانند مخففی که برای کلمه عمومی‌سازی ساخته شده، ایجاد شده است.

در شکل(۸-۲) مثال‌هایی از ساختار کدهای وابسته به زبان دیده می‌شود که در آن برنامه‌ها فقط برای محدوده جغرافیایی خاص خود قابل استفاده هستند.



شکل ۸-۲: ساختار کدهای وابسته به زبان

برای مثال یک بسته نرم‌افزاری حسابداری یا مالی که برای کشورهای آمریکا، کانادا، مکزیک و برزیل نوشته می‌شود، بایستی قابلیت بومی‌سازی برای پشتیبانی نوع نمایش و گزارشات، قوانین و پارامترهای مرتبط مانند تاریخ، زمان و پول هر کشور را داشته باشد. در واقع بومی‌سازی فرآیند سازگار کردن، ترجمه و بهینه سازی یک محصول (نرم‌افزار) برای یک بازار خاص است. در واقع بومی‌سازی تغییر دادن interface برای کاربران محلی، به صورتی که برای آنان قابل درک باشد؛ است.

موسسه استانداردهای بومی‌سازی (LISA)، بومی‌سازی را به این شکل تعریف می‌کند: "بومی‌سازی شامل متناسب ساختن زبانی و فرهنگی یک محصول برای یک محل مورد نظر (کشور / منطقه) است؛ به‌طوریکه آن محصول در آنجا فروخته شده و مورد استفاده قرار بگیرد." به‌طور کلی این تعریف شامل ترجمه user interface (پیغام‌هایی که به کاربر نمایش داده می‌شوند) برای ایجاد اطلاعات، تغییر اطلاعات، چاپ و ارسال آن به وسیله پست الکترونیکی و... می‌باشد.

به‌طور فنی بومی‌سازی نرم‌افزارهای متن‌باز با بومی‌سازی یک محصول تجاری، تفاوت خاصی ندارد. در هر دو حالت فونت‌ها باید عوض شوند؛ چیدمان کی بورد باید تغییر کند و استانداردها باید سازگار شوند. تنها تفاوت آنها در قیمت و licensing است. برای نرم‌افزارهای متن‌باز هزینه

کمتر و بحث licensing منتفی است. برای صرفه‌جویی در زمان و قیمت، پرورش نوآوری‌های محلی، و جلوگیری از کُپی غیرمجاز محصول، بومی‌سازی نرم‌افزارهای متن‌باز یک انتخاب بهتر است.

۸-۱-۳- اهمیت بومی سازی

در حال حاضر کسانی که بخواهند از کامپیوتر استفاده کنند؛ می‌بایست ابتدا زبان انگلیسی را آموزش ببینند. در کشوری با سطح سواد پائین، این موضوع دسترسی به تکنولوژی‌های اطلاعات و ارتباطات را، خصوصاً برای زنان و افراد فقیر که دسترسی مساوی به آموزش ندارند؛ محدود می‌کند. حتی پس از یادگیری زبان انگلیسی، کاربران می‌بایست صدها دلار برای گرفتن license نرم‌افزار بیگانه بپردازند و یا به کُپی غیرمجاز از نرم‌افزار متوسل شوند؛ تا بتوانند از این تکنولوژی‌ها استفاده نمایند. به‌طور خلاصه دسترسی به فناوری اطلاعات یکی از کلیدهای اصلی برای پیشرفت است و بوی‌سازی نرم‌افزارهای متن‌باز یک خلأ در این راه است.

بومی‌سازی فواید زیر را به دنبال دارد:

- به‌طور ویژه‌ای از میزان آموزش لازم به کاربران نهایی برای استفاده از سیستم‌های کامپیوتری می‌کاهد.
- راه را برای توسعه سیستم‌های کامپیوتری یک کشور، ایالت و محدوده باز می‌کند؛ تا برنامه نویسان محلی بتوانند به‌طور کامل بر روی زبان محلی و مدیریت پایگاه داده‌های محلی کار کنند.
- امکان تمرکززدایی داده‌ها را در سطوح محدود ایجاد می‌کند. این موضوع برای شرکت‌های خدمات‌رسانی مانند آب و برق و تلفن نیز صدق می‌کند؛ که می‌توانند پایگاه‌داده‌های با زبان محلی داشته و از هزینه‌های خود بکاهند و همچنین خدمات بهتری نیز به شهروندان ارائه کنند.
- به شهروندان اجازه می‌دهد تا از طریق پست الکترونیکی به زبان خود ارتباط برقرار کنند.
- به شرکت‌های نرم‌افزاری محلی اجازه می‌دهد تا برروی بومی‌سازی کار کنند.

ذینفعان این پروژه به شرح زیر هستند:

- به‌طور مستقیم، تمامی کاربران محلی که دسترسی آسانتری به کامپیوتر بدون فراگیری زبان انگلیسی پیدا خواهند کرد.
- به‌طور غیرمستقیم، تمامی شهروندان محلی که روابطشان با دولت تحت تاثیر قرار می‌گیرد.
- دولت محلی که فرصت توسعه پایگاه‌های‌داده و برنامه‌های کاربردی را به زبان محلی و بومی خود پیدا می‌کند. تکنولوژی مناسب و شرکت‌های نرم‌افزاری قدرتمند محلی به وجود خواهند آمد. دولت همچنین ابزار هماهنگی برنامه‌های کاربردی بین ایالات مختلف را نیز در اختیار خواهد داشت. در نتیجه توسعه‌های مبتنی‌بر IT در دولت می‌توانند با کمترین هزینه ممکن انجام شوند.
- صنعت نرم‌افزار. استفاده دولت از استانداردهای تکنولوژی کامپیوتر، شرکت‌های نرم‌افزاری را به توسعه سیستم‌های کامپیوتری رقیب که می‌توانند توسط دولت مورد استفاده قرار بگیرند، تشویق می‌کند. در نتیجه یک صنعت نرم‌افزاری با ثبات را در کشور ایجاد می‌نماید.

۸-۱-۴- دلایل بومی سازی نرم‌افزارهای متن باز

این یک واقعیت است که نرم‌افزارهای انگلیسی زبان نیازهای کشورها را برآورده نمی‌سازد. شرکت مایکروسافت و دیگر شرکت‌های آمریکایی بازار نرم‌افزارهای بین المللی را در دست دارند و سود زیادی را از این راه به دست آورده و می‌آورند. برای کشورهایی با منابع مالی محدود، پرداخت هزینه زیاد برای نرم‌افزارهای غیرمتن‌باز به معنی هزینه کمتر برای سایر برنامه‌های حیاتی آن کشور است.

زمانی که یک نرم‌افزار غیرمتن‌باز در یک محیط کاربرد فراوانی دارد، ترس از وابسته شدن به شرکت تولید کننده نرم‌افزار نیز وجود دارد. اگر شرکت تولید کننده نرم‌افزار را به زبان‌های دیگر پشتیبانی نکند، فقط کسانی که در انگلیسی خبره هستند می‌توانند از آن استفاده نمایند.

نکات مثبت کلیدی در مورد بومی‌سازی نرم‌افزارهای متن‌باز عبارتنداز :

- کاهش وابستگی به واردات
- عدم نیاز به یادگیری زبان انگلیسی برای کاربران بومی
- کسب تجربه برای برنامه‌نویسان بومی

- کنترل بومی بر قابلیت‌ها و ظاهر نرم‌افزار
- پایه‌گذاری صنعت نرم‌افزار محلی
- نکات منفی استفاده از نرم‌افزارهای غیرمتن‌باز :
- گرانی license و نگهداری آنها
- پیاده‌سازی شده با زبان انگلیسی
- کنترل توسط شرکت‌های خارجی
- وابسته به استانداردهای غیرمتن‌باز یا محرمانه
- عدم وجود پشتیبانی درست
- محدود کردن صنعت نرم‌افزار محلی
- نداشتن قابلیت تغییر نرم‌افزار توسط برنامه‌نویسان بومی

نرم‌افزارهای متن‌باز عموماً ایمن‌تر از نرم‌افزارهای غیرمتن‌باز هستند. برای تضمین امنیت ملی، باید از نرم‌افزارهایی استفاده کرد که وابسته به کشورهای دیگر نباشند. سیستم‌های متن‌باز به ملت‌ها و شهروندان اجازه می‌دهد تا خودشان کدهای برنامه را متناسب با نیازها و ویژگی‌های منطقه‌ای خود تغییر داده و به دنبال راهکارهایی برای خود باشند.

در حال حاظر اکثر زبان‌هایی که برای برنامه‌سازی در سیستم‌عامل Multos استفاده می‌شوند امکانات و قابلیت‌های خوبی برای I18N برخوردار هستند.

۸-۲- اجرای بومی سازی

در فرآیند بومی‌سازی یک سیستم، بیشترین سهم را برنامه‌نویسان برعهده دارند. آنان در این پروسه نیازمند کمک مترجمان، نویسندگان و تست‌کنندگان هستند. قبل از آغاز بومی‌سازی یک نرم‌افزار، تهیه لغت‌نامه و استانداردهای لازم ضروری می‌باشد. دربومی‌سازی در سطح بین‌المللی، بخصوص زمانیکه زبان‌ها با یکدیگر وجوه مشترک دارند (مانند فارسی و عربی و یا چینی و ژاپنی و کره‌ای– CJK)، به اشتراک گذاشتن منابع فنی و برنامه‌نویسی می‌تواند بسیار کمک‌کننده و موثر باشد.

کشورهای آسیایی می‌توانند روشی را که CJK (سه کشور چین, ژاپن و کره) برای به اشتراک گذاشتن منابع خود استفاده کردند، دنبال نمایند. اگرچه در هر صورت پیاده‌سازی و اجرای بومی‌سازی هزینه‌های خاص خود را برای آنان دربر خواهد داشت. فرآیند بومی‌سازی باید

اهداف واضح و مشخصی داشته باشد و منابع لازم را برای رسین به آن اهداف تهیه کند. این فرآیند نیازمند منابع مالی، مدیریت حرفه‌ای و پیشرفته و تجارب فنی است. علاوه بر این دانش زبان‌شناسی نیز مورد نیاز است.

برپایی مراکز بومی‌سازی برای متمرکز شدن برنامه نویسان جهت تبادل اطلاعات، گسترش مهارت‌ها و تکمیل موفقیت آمیز فعالیت‌ها مورد نیاز است. زمان یکی دیگر از مسائل مهم در بومی‌سازی است؛ زیرا اگرچه صحت انجام کارها از اهمیت زیادی برخوردار است، اما سرعت انجام آنها نیز بسیار مهم است. برای مثال ایجاد یک فرهنگنامه تخصصی کامپیوتر به زبان ساده می‌تواند بیش از یک سال طول بکشد؛ ولی یک لغتنامه اولیه نیز می‌تواند برای ترجمه نسخه‌های اول برنامه کافی باشد. نسخه‌های بعدی می‌توانند از لغتنامه کامل استفاده کنند.

با توجه به مسائل فوق، اهمیت نقش دولت در بومی‌سازی نرم‌افزارها مشخص می‌شود. بدون پشتیبانی دولت، توسعه صنعت بومی‌سازی نرم‌افزارهای متن‌باز با مشکلات جدی روبه‌رو است.

۸-۲-۱- مهارت‌ها و ابزار مورد نیاز جهت بومی سازی

بومی‌سازی زمانی اتفاق می‌افتد که کشور در حال استفاده از سیستم‌های کامپیوتری است، اما سیستم‌هایی به زبان بیگانه. دانشمندان علوم کامپیوتر و یا دانشجویان عموماً از لغات انگلیسی یا فرانسه در کارهای تخصصی استفاده می‌کنند ولی کاربران عادی ممکن است با زبان بیگانه‌ای آشنایی نداشته باشند. بنابراین بومی‌سازی نیاز به فرهنگنامه‌ای برای تبدیل لغات بیگانه به بومی دارد تا کاربران ابتدایی بتوانند از زبان محلی خود استفاده کنند.

پس از آن نوبت به کار همزمان مترجمان و متخصصین کامپیوتر در یک گروه می‌رسد تا هم از لحاظ فنی و هم از لحاظ زبان‌شناسی، نرم‌افزار را بومی نمایند.

در واقع مهارت‌های مربوط به پروژه بومی‌سازی به دو بخش تقسیم می‌شود:

- بخش فنی که اصلاح کدهای برنامه را هدایت می‌کند.
- بخش ترجمه که فعالیت‌های مربوط به زبان‌شناسی و نوشتاری را هماهنگ می‌کند.

ابزارها و امکانات لازم برای بومی‌سازی نرم‌افزارهای متن‌باز کم هزینه‌تر از نرم‌افزارهای غیرمتن‌باز است. تجربه نشان می‌دهد که استفاده از ابزارهای متن‌باز و بومی‌سازی آنها بهترین راه‌حل است.

۸-۲-۲- هزینه‌های بومی سازی نرم‌افزارهای متن باز

محاسبه هزینه‌ها و تخمین آن، فرآیند ساده‌ای نیست. علاوه بر هزینه ترجمه واژگانی که در بومی‌سازی نیاز به تغییر دارند، عوامل و هزینه‌های زیر نیز بایستی درنظر گرفته شوند:

تجربه و مهارت: آیا برنامه‌نویسان، مترجمان و دیگر افراد پروژه سابقه‌ای در زمینه این نوع کار دارند؟ اگر نه، زمان و تلاش بیشتری برای آموزش فرآیندها و استانداردهای بومی‌سازی به آنها لازم است.

محیط کاری : آیا افراد پروژه ابزارها و امکانات لازم برای انجام کار به صورت حرفه‌ای را در اختیار دارند؟ بدون وجود محل کار مناسب ابزارها و تکنیک‌ها، انتظار انجام پروژه در بالاترین کیفیت غیرواقعی است.

عوامل زبان شناسی: زبان بومی موردنظر با زبان انگلیسی چقدر تفاوت دارد؟ برای مثال ترجمه از زبان انگلیسی به زبان سوئدی کار ساده‌ای است. گرامر، لغات و طول لغات بسیار نزدیک به هم است. در حالیکه ترجمه از زبان انگلیسی به زبان تایلندی بسیار مشکل است. گرامر، املا، طول لغات، مقایسه دستخط‌ها و ... به‌طور کلی با هم متفاوت است. بنابراین اندازه و محل المان‌های فرم‌ها بایستی تغییر کنند. علاوه بر این نبودن مترجمان متبحر و همچنین لغتنامه تخصصی این روند را بسیار کندتر می‌کند.

حدود: چقدر کافی است؟ آیا تغییر منوهای ابتدایی نیز مورد نیاز است؟ آیا فایل‌های help نیز باید ترجمه شوند؟ و ... برای جلوگیری از شکست در پروژه باید تعریف جامعی از حدود و مرزهای پروژه مشخص شود.

آمار و ارقام: در پروژه‌های پیشرفته نرم‌افزاری تخمین هزینه‌های مالی و زمانی بسیار مهم است و این تخمین با استفاده از هزینه پروژه‌های قبلی دقیق‌تر می‌شود. بنابراین داشتن آرشیو کاملی از نفرساعت‌ها، منابع، تجارب و سایر موارد می‌تواند کمک موثری برای آینده باشد.

۸-۲-۳- نکات کلیدی بومی سازی سیستم عامل‌های کارت هوشمند

در این بخش به نکات کلیدی که برای بومی‌سازی سیستم‌عامل‌های کارت‌های هوشمند لازم است، اشاره می‌شود.

۸-۲-۳-۱- استانداردسازی

نرم‌افزارها برای یکپارچگی و سهولت استفاده نیاز به پیروی از استانداردهای خاص دارند. در مهندسی نرم‌افزار استانداردسازی جز اولین قدم‌های اجرای پروژه می‌باشد. برای شروع بومی‌سازی سیستم‌عامل‌های کارت هوشمند می‌توان از استانداردهای موجود و مرتبط در این زمینه استفاده کرد. در حال حاضر استانداردهای بین‌المللی زیادی برای پوشش دادن کلیه زبان‌های موجود در دنیا تهیه شده‌است. مهم‌ترین این منابع عبارتنداز :

ISO/IEC JTC1
Unicode Consortium
Free Standards Group

توجه داشته باشید که به بعضی نکات، مانند نقشه‌های صفحه کلید (keyboard maps) و یا متدهای ورودی/خروجی، در استانداردهای فوق اشاره‌ای نشده است. استانداردهای ملی می‌بایست این نکات را متناسب با نیازهای خود تعیین کنند.

۸-۲-۳-۲- Unicode

کاراکترها مهم‌ترین و اصولی‌ترین بخش برای نمایش متن هر زبانی هستند. در دهه ۱۹۷۰، برنامه‌نویسان از کاراکترستی متشکل از الفبای انگلیسی، ارقام اعشاری و علائم علامت‌گذاری استفاده می‌کردند. رایج‌ترین کاراکتر ستی که مورد استفاده قرار گرفت ASCII هفت بیتی بود که قابلیت نمایش ۱۲۸ کاراکتر را داشت و تنها برای زبان انگلیسی کافی بود. با گسترش استفاده از زبان‌های غیرانگلیسی، نیاز به استفاده از کاراکتر ست‌های جدید نیز احساس شد. کاراکتر ست‌های موجود پاسخگوی نیازهای زبان‌های آسیایی نبود و لذا کنسرسیوم Unicode برای دربرگرفتن کلیه زبان‌ها ایجاد شد.

امروزه بسیاری از برنامه‌های کاربردی به سمت استفاده از Unicode حرکت کرده‌اند تا از زبان‌های جدید نیز پشتیبانی کنند. برای مثال کاراکترست سیستم عامل Multos بر مبنای یونی‌کد (Unicode) است، بدین منظور نوع داده‌ای char برای تطابق با یونی‌کد، دو بایتی (۱۶ بیت) در نظر گرفته شده است. بنابراین نوع داده‌ای String که با استفاده از char ساخته می‌شود نیز یونی‌کد را پشتیبانی می‌کند. یونی‌کد طوری تعریف شده است که مقادیر ۰ تا ۱۲۷ برای استاندارد ASCII و ۰ تا ۲۵۵ برای استاندارد ISO 8859-1 (Latin-1) را در برمی‌گیرد. به دلیل شروع شدن دو استاندارد با عدد صفر، برنامه‌نویسانی که از قابلیت‌های I18N استفاده نمی‌کنند یا آشنا نیستند نیز می‌توانند بدون نیاز به شناخت یونی‌کد، برنامه‌هایی بنویسند که

یونی‌کد را پشتیبانی کند، اما برنامه‌نویسانی که در محیط ویندوز برنامه می‌نویسند بایستی از تفاوت‌های استاندارد ISO 8859-1 و Windows Latin-1 (CP1252) آگاهی داشته باشند .طول ۱۶ بیتی نوع داده‌ای char امکان ذخیره سازی مقادیر ۰ تا ۶۵۵۳۵ را فراهم می‌سازد. یک مقدار یونی‌کد با '\u' شروع و یک مقدار هگزادسیمال (مبنای ۱۶) از ۰۰۰۰ تا FFFF پس از آن قرار می‌گیرد. دو خط مثال زیر، کاراکتر a را تعریف می‌کند:

```
char c1 = 'a'
char c2 = '\u0061'
```

نسخه JDK 1.3 یونی‌کد ۲.۱ و نسخه JDK 1.4 یونی‌کد ۳.۰ را پشتیبانی می‌کند .

سؤالی که مطرح می‌شود این است که آیا همه پلاتفرم‌ها یونی‌کد را پشتیبانی می‌کنند؟ جواب: در جاوا تمام استریم‌ها (Stream) که کاراکترها را پشتیبانی می‌کنند java.io.Reader) و java.io.Writer)به صورت اتوماتیک یک لایه مخفی که وظیفه تبدیل یونی‌کد به Encoding خاص سیستم‌عامل و بالعکس را انجام می‌دهند را فراخوانی می‌سازند. از طرف دیگر کلاس‌های java.io.InputStreamReader و java.io.OutputStreamWriter دارای متدهایی برای تبدیل Encodingمی‌باشند.

Locale –۳–۳–۲–۸

Locale در سیستم‌عامل Multos مجموعه‌ای از کلاس‌ها برای سفارشی کردن، تغییر، نمایش و شکل دهی اطلاعات است. این کلاس‌ها روی انتخاب زبان، تقویم، تاریخ و زمان تأثیر می‌گذارند. یک شیء از کلاس Locale نشان‌دهنده یک ناحیه جغرافیائی خاص با زبان آن ناحیه می‌باشد.

Language –۴–۳–۲–۸

زبان‌های قابل استفاده در Locale طبق استاندارد ISO 639 می‌باشد. جدول(۸-۱) چند مثال از نام زبان و کد تعریف شده را نشان می‌دهد. یک کد زبان با جزئیات آن زبان در نواحی مختلف استفاده نمی‌شود، برای مثال ممکن است Canadian French و Swiss French دارای گرامر و قوانین و واژه‌های متفاوت باشند ولی برای هر دو این زبان‌ها کد fr درنظر گرفته می‌شود. به همین دلیل تنها زبان‌های عمومی تعریف شده‌اند.

جدول ۸-۱: مثال‌هایی از زبان‌های قابل استفاده در Locale

زبان	کد زبان
English	en
French	fr
Chinese	zh
Japanese	ja
Arabic	ar

Country–۵–۳–۲–۸

کشورهای قابل استفاده در Locale طبق استاندارد ISO 3165 می‌باشد. کد کشور در این استاندارد ۲ رقمی و با حروف بزرگ تعریف شده است. جدول(۸-۲) زیر تعدادی از کشورهای تعریف شده را نشان می‌دهد

جدول ۸-۲: کشورهای تعریف شده در Locale

کشور	کد کشور
United States	US
France	FR
Canada	CA
Saudi Arabia	SA

Variant–۶–۳–۲–۸

Variantیک پسوند اختیاری برای یک Locale می‌باشد. Variant یک Locale سفارشی تعریف می‌کند که با Language و Country قابل ساخت نیست. از Variant می‌توان برای اضافه کردن یک مشخصه اضافی برای تعریف Locale استفاده کرد. برای مثال en _ us نشان دهنده English (United States) است اما en_us_ca برای تعریف English (U.S.A, California) یا en_us_mac برای تعریف English (U.S.A, Macintosh) می‌باشد.

کلاس Locale دو سازنده (Constructor) دارد :

```
Locale (String language, String country)
Locale (String language, String country, String variant)
```

مثال :

```
Locale mylocale = new Locale ("en", "US")
```

در مثال بالا en زبان انگلیسی و US کشور آمریکا را تعریف می‌کند.

Number –۷–۳–۲–۸

عبارت ۱,۲۳۴ چه عددی را نشان می‌دهد؟ جواب به منطقه و کشور بستگی دارد. در آمریکا این عبارت معنی "یک هزار و دویست و سی و چهار" را دارد اما در فرانسه این عبارت به معنی "یک و دویست و سی و چهار هزارم" است. بنابراین در نظر گرفتن قوانین نمایش اعداد و جدا کننده‌ها بسیار مهم بنظر می‌رسد .کلاس java.text.NumberFormat تعریف چگونگی نمایش اعداد برای یک منطقه خاص را به‌عهده دارد.

Currency–۸–۳–۲–۸

هر کشور یا منطقه دارای واحد پولی جداگانه است .علاوه بر این مواردی از قبیل علامت اعداد منفی، صفر، جداکننده رقم‌ها، علامت نقطه اعشاری و مکان قرارگرفتن علامت پول($۱۰،۱۰ریال) نیز در کشورها متفاوت است. متدgetCurrencyInstance درکلاس java.text.NumberFormat امکان پرداختن به جزئیات شکل دهی اعداد پولی یک کشور خاص را فراهم می‌سازد .

Date –۹–۳–۲–۸

مشابه سایر ساختارهای حساس به موقعیت جغرافیایی، فیلد تاریخ نیز جزئیات خاص خود را دارد. کوتاه یا کامل بودن تاریخ، جدا کننده روز، ماه و سال و نیز ترتیب آنها در کشورهای مختلف با یکدیگر متفاوت است.

Calendar –۱۰–۳–۲–۸

کلاس java.text.Calendar بسیار نزدیک به کلاس java.util.Date است و اجازه می‌دهد سال، ماه، روز و زمان را از یک تاریخ استخراج نمایند. متد getCalendarInstance یک Calendar برای کشور مورد نظر برمی‌گرداند. تنها تقویم پیاده‌سازی توسط شرکت سان میکرو سیستم تقویم Gregorian می‌باشد. برای ساختن تقویم‌های محلی توصیه می‌شود از کلاس Calendar جاوا استفاده شود.

Resource Bundle –۱۱–۳–۲–۸

ویژگی عمومی ساختن درJDK ، مکانیزمی برای جداسازی عناصر رابط کاربر و سایر اطلاعات حساس به منطقه جغرافیایی از منطق برنامه است. این جداسازی انتقال فرآیند و تبدیل را آسان می‌سازد. بدین معنی که یک کد واحد نوشته می‌شود، در حالیکه ممکن است ۳۰ نسخه به زبان‌های مختلف از آن تولید شود .

Resource Bundleشامل یک فایل با فرمت ASCII است. نام این فایل شامل دو قسمت، نام اصلی و یک پسوند است. برای مثال برای ایجاد یک Resource Bundle با نام MyBundle برای زبان‌های ژاپنی و فرانسوی دو فایل بنام‌های MyBundle_ja_JP و MyBundle_fr_FR با پسوند properties تعریف می‌شود. هر فایل Resource Bundle شامل یک‌سری زوج کلید/ مقدار است. در هر خط بایستی یک زوج کلید/ مقدار تعریف شود. مثال :

```
#MyResource.properties
#<key>=<value>
Text_not_found=The file could not be found
Text_Hello=Hello, world!
```

متد getBundle در کلاس java.util.ResourceBundle امکان دسترسی به یک Resource Bundleرا فراهم می‌سازد. مثال کاملی از تعریف Locale برای زبان فارسی در انتهای این مقاله ذکر شده است.

Character Sets –۱۲–۳–۲–۸

زبان جاوا با استفاده از یونی‌کد برای نمایش متن، فرآیند ذخیره سازی، دستکاری و نمایش کاراکترها را آسان ساخته است. یونی‌کد یک کارکترست ۱۶ بیتی است بدین معنی که می‌تواند ۱۶۲ کاراکتر تعریف کند. هر کاراکتر در این مجموعه منحصر بفرد است.

Layout Manager –۱۳–۳–۲–۸

Layout Manager در یک برنامه چند زبانه بدلیل دو مشکل اساسی در هنگام ترجمه رابط کاربر بسیار مهم است :

- افزایش و کاهش طول متن ترجمه شده
- موقعیت اجزاء رابط کاربر

قبل ازInternationalization : در نظر بگیرید که یک برنامه نوشته اید که چند پیغام را نمایش می‌دهد :

```
public class NotI18N {
   static public void main(String[] args) {
      System.out.println("Hello.");
      System.out.println("How are you?");
      System.out.println("Goodbye.");
   }
}
```

حال شما تصمیم گرفته اید که برنامه را طوری تغییر دهید که پیغام‌ها را به زبان‌های فرانسه، آلمانی و فارسی نمایش دهد. متاسفانه تیم برنامه‌نویس با زبان‌های دیگر آشنایی چندانی ندارد، بنابراین شما به یک مترجم برای ترجمه پیغام‌ها به زبان‌های ذکر شده نیاز دارید. از آنجایی‌که مترجم شما برنامه‌نویس نیست، شما مجبور هستید این پیغام‌ها را از داخل کد برنامه خارج و در یک فایل متن ذخیره نمایید تا مترجم آن را ترجمه کند. علاوه‌بر این برنامه بایستی انعطاف‌پذیر باشد و قابلیت اضافه کردن یک زبان جدید را نیزداشته باشد.

بعد ازInternationalization کد زیر تغییر یافته برنامه شما را نشان می‌دهد. توجه کنید که متن پیغام‌ها از داخل کد برنامه خارج شده است :

```
import java.util.*;
public class I18NSample {
   static public void main(String[] args) {
      String language;
      String country;
      if (args.length != 2) {
         language = new String("en");
         country = new String("US");
      } else {
         language = new String(args[0]);
         country = new String(args[1]);
      }
      Locale currentLocale;
      ResourceBundle messages;
      currentLocale = new Locale(language, country);
      messages = ResourceBundle.getBundle("MessagesBundle", currentLocale);
      System.out.println(messages.getString("greetings"));
      System.out.println(messages.getString("inquiry"));
      System.out.println(messages.getString("farewell"));
   }
}
```

برای کامپایل و اجرای برنامه به فایل‌های زیر نیاز دارید :

```
-I18NSample.java
-MessageBundle.properies
-MessageBundle_de_DE.properties
-MessageBundle_en_US.properties
-MessageBundle_fr_FR.properties
-MessageBundle_ar_SA.properties
```

برنامه عمومی شده انعطاف‌پذیر است. این برنامه به کاربر نهایی اجازه می‌دهد کشور و زبان را بصورت پارامتر به برنامه وارد کند. مثال زیر نتیجه اجرای برنامه را برای زبان فرانسوی (French) fr و کشور فرانسه (France) FR نشان می‌دهد :

```
greetings = Bonjour
farewell = Au revoir
inquiry = Comment allez-vous?
فایل MessageBundle_fr_FR.properties
% java I18NSample fr FR
Bonjour.
Comment allez-vous?
Au revoir.
```

مثال بعدی نتیجه اجرای برنامه را برای زبان انگلیسی (English) en و کشور آمریکا (United States) US نشان می‌دهد :

```
greetings = Hello
farewell = Goodbye
inquiry = How are you?
فایل MessageBundle.properties
    % java I18NSample en US
Hello.
How are you?
Goodbye.
```

۸-۲-۴- اقدامات لازم برای I18N

بسیاری از برنامه‌ها از ابتدا عمومی و چند زبانه نیستند. این برنامه‌ها ممکن است از یک Prototype شروع شوند و یا قابلیت چند زبانه بودن در آنها درنظر گرفته نشده باشد. برای I18N ساختن یک برنامه موجود، بایستی مراحل زیر طی شده باشد.

۸-۲-۴-۱- شناسایی اطلاعات حساس به زبان و منطقه

پیغام‌های متنی صریح‌ترین فرم اطلاعاتی هستند که به زبان و منطقه جغرافیایی وابستگی دارند. به هرحال سایر اطلاعات نیز ممکن است به این دو عامل وابستگی داشته باشند .لیست زیر اطلاعات حساس به ملیت را نشان می‌دهد :

پیغام‌ها-عنوان اجزای رابط کاربر-صداها-رنگ‌ها-گرافیک و تصاویر-تاریخ-زمان-پول-واحدهای اندازه‌گیری-شماره تلفن-آدرس‌های پستی-طراحی صفحات.

۸-۲-۴-۲- جداسازی متن‌های قابل ترجمه و انتقال به Resource Bundles

فرآیند ترجمه هزینه‌بر و گران است. شما می‌توانید این هزینه را با جداسازی متن‌های نیازمند ترجمه و قرار دادن آنها در یک Resource Bundle کاهش دهید .

متن‌های قابل ترجمه شامل پیغام‌های وضعیت، پیغام‌های خطا، فایل‌های log و عنوان اجزا GUIمی‌باشند. ضمن جداسازی این اطلاعات از کد برنامه، متغیرهایی که از این عنوان‌ها استفاده می‌کنند بایستی به‌طور صریح و جداگانه تعریف شوند :

```
String buttonLabel = "OK";
JButton okButton = new JButton(buttonLabel);
```

۸-۲-۴-۳- پیغام‌های ترکیبی

پیغام‌های ترکیبی حاوی اطلاعات متغیر است. برای مثال در پیغام "The disk contains 1100 files" عدد ۱۱۰۰ می‌تواند متغیر باشد. ترجمه این نوع پیغام بسیار مشکل است، زیرا برای مثال محل قرارگرفتن عدد ۱۱۰۰ در این عبارت در زبان‌های مختلف با یکدیگر متفاوت است. پیغام زیر قابل‌ترجمه نیست چون ترتیب قرار گرفتن قسمت‌های این جمله در داخل کد برنامه مشخص شده است.

```
Integer fileCount;
...
String diskStatus = "The disk contains " + fileCount.toString() + " files.";
```

۸-۲-۴-۴-نمایش اعداد و پول

اگر برنامه اعداد یا پول را نمایش می‌دهد، بایستی این اطلاعات به روشی مستقل از کشور یا منطقه خاص نمایش داده شود. کد زیر عمومی نیست چرا که نمی‌تواند اعداد را بصورت صحیح در تمام کشورها نمایش دهد :

```
Double amount;
TextField amountField;
...
String displayAmount = amount.toString();
amountField.setText(displayAmount);
```

در جاوا کلاس NumberFormat برای تغییر نمایش اعداد به زبان‌های مختلف در نظر گرفته شده‌است.

۸-۲-۴-۵- نمایش تاریخ و زمان

نمایش تاریخ و زمان نیز مشابه اعداد با توجه به زبان و کشور متفاوت است. اگر شما کدی مشابه زیر در برنامه دارید، بایستی آنرا تغییر دهید :

```
Date currentDate = new Date();
TextField dateField;
...
String dateString = currentDate.toString();
dateField.setText(dateString);
```

در زبان جاوا کلاس DateFormat وظیفه تنظیم چگونگی نمایش تاریخ و زمان را برعهده دارد .

۸-۲-۴-۶- استفاده از قابلیت‌های یونی کد

کد زیر سعی می‌کند کاراکتر بودن یک حرف را بررسی کند :

```
char ch;
...
if ((ch >= 'a' && ch <= 'z') ||
   (ch >= 'A' && ch <= 'Z'))      // WRONG!
```

این کد تنها برای زبان انگلیسی قابل استفاده است. برای مثال این کد، حرف ü در کلمه آلمانی Grün یا حرف پ در کلمه پارس را تشخیص نمی‌دهد .

متدهای مقایسه موجود در کلاس Character از استاندارد یونی‌کد برای شناسایی کاراکترها استفاده می‌کند. بنابراین کد بالا را می‌توان بصورت زیر تغییر داد:

```
char ch;
...
if (Character.isLetter(ch))
```

۸-۲-۴-۷- مقایسه رشته ها

معمولا هنگام مرتب‌سازی متن، رشته‌ها با یکدیگر مقایسه می‌شوند. یک برنامه معمولی برای مقایسه دو رشته ممکن است از کدی به صورت زیر استفاده کند:

```
String target;
String candidate;
...
if (target.equals(candidate)) {
...
if (target.compareTo(candidate) < 0) {
…
```

متدهای String.equals و String.compareTo عمل مقایسه را بصورت باینری انجام می‌دهند که برای بیشتر زبان‌های موجود ناکارآمد است. برای مقایسه دو رشته بهتر است از کلاس Collator استفاده شود.

۸-۲-۴-۸-تبدیل متن‌های غیر یونی کد

کاراکترها در زبان برنامه نویسی جاوا بصورت یونی‌کد ذخیره می‌شود. اگر برنامه شما از اطلاعات غیر از یونی‌کد استفاده می‌کند، بایستی اطلاعات را به یونی‌کد تبدیل نمایید .

۸-۲-۵- فارسی سازیMULTOS

۸-۲-۵-۱- تاریخ هجری شمسی

تقویم هجری شمسی در ایران و نواحی اطراف مانند افغانستان، جمهوری‌های آسیای مرکزی و کردهای بین النهرین به طور رسمی استفاده می‌گردد. تقویم جلالی بخاطر اقدام جلال الدین ملکشاه سلجوقی که توسط شاعر و ریاضیدان بزرگ ایرانی عمر خیام در اواخر قرن پنجم به تصحیح تقویم هجری شمسی همت گمارد، نامگذاری شد. تقویم گریگوری نیز از نام پاپ گریگوری هشتم که آخرین تغییرات را در سال ۱۵۸۲ ب.م. در تقویم میلادی ایجاد کرد، گرفته شده است. طول سال در این دو تقویم کم و بیش به یک اندازه است، اما براساس روش‌های کاملا متفاوتی پایه گذاری شده‌اند .

تقویم جلالی دارای خصوصیتی طبیعی و کلی است. این تقویم تمام چهار فصل سال را با آغازی مصادف با اولین روز بهار (در نیمکره شمالی) در برمیگیرد. همچنین این تقویم دارای یک قانون منظم (نه یکنواخت) برای روزهای ماه است. شش ماه اول هر سال ۳۱ روز و شش ماه دوم در سال‌های کبیسه ۳۰ روز می‌باشد، در غیر اینصورت ماه آخر سال ۲۹ روزه خواهد بود. تقویم جلالی تقریبا" با سیستم رایج زودیاک (Zodiac System) غرب تطابق دارد .

در این دو مورد تقویم گریگوری با تقویم جلالی دارای اختلافاتی است. در این تقویم ارتباطی بین سال و فصل‌ها وجود ندارد. به‌طور قراردادی، تعداد روزها در هفت ماه سال ۳۱ روز، در چهار ماه ۳۰ روز و یک ماه در سال به‌طور نامنظم ۲۸ یا ۲۹ روز است.

علاوه‌بر این، تقویم جلالی از حرکت وضعی زمین (حرکت زمین به دور خورشید) تبعیت می‌کند. یک سال خورشیدی تقریبا معادل ۳۶۵ روز، ۵ ساعت و ۴۹ دقیقه است. براساس تقویم

جلالی، آغاز سال (تحویل سال) زمانی است که خورشید از نیمکره جنوبی به سمت نیمکره‌شمالی از روی خط استوا می‌گذرد. حال اگر لحظه تحویل سال قبل از ظهر (به وقت تهران) باشد، سال جدید (اول فروردین ماه) از همان روز آغاز می‌شود. در غیراینصورت، آغاز سال نو روز بعد خواهد بود. در این روش (به طور تقریبی) در هر ۳۳ سال، ۸ سال کبیسه تعیین می‌گردد. سال‌های کبیسه، سال‌هایی هستند که پس از تقسیم بر ۳۳ باقیمانده آنها ۶، ۲۲، ۱۷، ۱۳، ۹، ۵، ۱ و ۳۰ باشد. به‌عنوان مثال، چنانچه ۱۳۷۰ را به ۳۳ تقسیم کنیم، باقیمانده ۱۷ خواهد بود پس سال ۱۳۷۰ سال کبیسه است. سال کبیسه بعدی سال ۱۳۷۵ می باشد با چهار فاصله– در صورتیکه به طور معمول فاصله بین دو سال کبیسه ۳ سال است.

تقویم گریگوری از روش ساده‌ای استفاده کرده که دارای اختلافاتی با سال طبیعی می‌باشد. در این روش هر چهار سال یک‌بار سال کبیسه خواهد بود، مگر سال‌هایی که قابل قسمت به۱۰۰ باشند، اما بر۴۰۰ نباشند. در نتیجه سال ۲۰۰۰ سال کبیسه است (از آنجایی که بر۴۰۰ قابل قسمت است) اما سال ۲۱۰۰ کبیسه نیست.

مواردیکه در مبحث تاریخ هجری شمسی بایستی در نظر گرفته شوند عبارتنداز :

- محاسبه سال‌های کبیسه
- تبدیل تاریخ میلادی به تاریخ هجری شمسی
- تبدیل تاریخ هجری شمسی به تاریخ میلادی
- نمایش تاریخ شمسی به شکل‌های مختلف (کوتاه، کامل)
- تابع چک کننده تاریخ هجری شمسی
- Persian Date Taglib

۸-۲-۵-۲- تقویم هجری شمسی

مواردیکه در مبحث تقویم هجری شمسی بایستی لحاظ شود عبارتستاز:

- تفکیک مقادیر سال، روز، ماه برای تاریخ تعیین شده
- مشخص کردن روز هفته/ اولین روز هفته/ اولین روز ماه
- قابلیت تغییر تاریخ با متدهای roll و add
- TaglibPersian Calendar

مراجع

[1] Operating Systems Design and Implementation, Third Edition, By Andrew S. Tanenbaum.
[2] Operating Systems Concepts, 6th Edition, Abraham Silberschatz.
[3] Modern Operating Systems, By Andrew S. Tanenbaum.
[4] Distributed Operating Systems, By Andrew S. Tanenbaum.
[5] The Open Group Base Specifications Issue 6 , IEEE Std 1003.1, 2004 Edition, opengroup.org
[6] Linux Standard Base (LSB), freestandards.org
[7] http://www.Opengroup.org (Unix Trade Mark Owner)
[8] http://www.bell-labs.com/history/unix (Unix History)
[9] http://www.levenez.com/unix (Unix History)
[10] http://www.linux.org/ (Linux Home Page)
[11] http://www.freebsd.org (FreeBSD Project)
[12] http://www.opensolaris.org (OpenSolaris Project)
[13] http://www.freebsdsoftware.org (FreeBSD software and Port Archive)
[14] http://www.trustedbsd.org (TrustedBSD Project)
[15] http://www.netbsd.org (NetBSD Project)
[16] http://www.openbsd.org (OpenBSD Project)
[17] http://www.dragonflybsd.org (Dragonfly BSD)
[18] http://www.bsdportal.org (BSD Portal, BSD Family News and Resources)
[19] http://www.bsdcertification.org (BSD Certification Group)
[20] http://plan9.bell-labs.com/plan9 (Plan 9)
[21] http://www.opendarwin.org (OpenDarwin Project)
[22] http://www.vitanuova.com/inferno (Inferno Project)
[23] http://www.reactos.org (ReactOS Project)
[24] http://www.freedos.org (FreeDOS Project)
[25] http://www.centos.org (Centos Linux Distributor)
[26] http://www.debian.org (Debian Linux Distributor)
[27] http://www.redhat.com (Red Hat Linux Distributor)
[28] http://en.wikipedia.org/wiki/List_of_operating_systems (Great List Of Operating Systems)
[29] http://www.opensource.org/licenses/bsd-license.php (BSD License)
[30] http://www.gnu.org/copyleft/gpl.html (GPL License)
[31] http://www.nsa.gov/selinux/
[32] http://www.novell.com/linux/ (SUSE Enterprise Linux)
[33] http://www.ubuntu.com/ (Ubuntu Linux)
[34] Real-Time Systems Design And Analysis, Phillip A. Laplante.
[35] Real-Time Concepts for Embedded Systems, by Qing Li and Carolyn Yao.
[36] http://www.freestandards.org (Linux Standard Base , LSB)
[37] http://en.wikipedia.org/wiki/Comparison_of_operating_systems (Comparison of many operating systems.

[38] Anousak Souphavanh – Theppitak Karoonboonyanan.
[39] Asia Pacific Development Information Program (APDIP) – 2005.
[40] Guide to Smart Card Technology. Samuel Chanson- Cuyberspace Center.
[41] Java Card: An analysis of the most smart card operating system to date . Eduardo de Jong, Pieter Hartel, Patrice Peyret, Peter Catteneo – 2005.
[42] http://developers.sun.com/techtopics/mobility/javacard/articles/javacard2/
[43] http://java.sun.com/products/javacard/
[44] MULTOS Technical Paper. Brian McKeon, Raymond Makewell.
[45] Architecture of a portable Smart Card Operating System .Felix Pletzer – 2004.
[46] Introduction to Smart Cards . Sumit Dhar - Auerbach Publications – 2003.
[47] http://java.sun.com/products/javacard/
[48] http://www.microsoft.com/windowsce/smartcard/
[49] An Introduction to Java Card Technology - Part 2, The Java Card Applet. C. Enrique Ortiz - September 2003.
[50] MAOSCO Consortium, 2000. MULTOS: The Multi-Application Operating System for Smart Cards.
[51] Smart Cards and their Operating Systems .Heng Guo - HUT, Telecommunications Software and Multimedia Laboratory – 2001.