مدل توسعه هسته لینوکس

تهیه کننده : ایمان دارابی

Iman.darabi@gmail.com

پاییز ۸۷

چکیده

اوایل دوران توسعه نرم افزارها دارا بودن توانایی اولیه برنامه نویسی برای نوشتن پروژه های نرم افزاری کافی بود . امروزه با افزایش حجم برنامه ها و نیاز به همکاری در برنامه نویسی مورد توجه قرار گرفته است .

سیستم عامل لینوکس با بهره گیری از اینترنت و مدل توسعه غیر متمرکز توانسته تبدیل به یکی از موفق ترین پروژه های متن باز در دنیا شود. به دلیل ارتباط تنگاتنگ میان برنامه نویسان و کاربران در این مدل توسعه کیفیت و کارایی نرم افزار ها بویژه سیستم عامل لینوکس افزایش یافته است. همچنین برای افزایش ارتباط بین این دو گروه چرخه ارتقاع نرم افزارها از بازه زمانی بلند مدت به بازه های کوتاه مدت زمانی تغییر کرده است

مقدمه :

بی گمان دهه های گذشته داشتن اطلاعات در زمینه زبانهای برنامه نویسی و توانایی کار کردن با کامپیوتر ها کافی بود تا بتوان نرم نرم افزاری ایجاد نموده و آنرا توسعه داد. در آن زمان منطق برنامه ها به پیچیدگی برنامه های امروزی نبوده و برنامه ها ندرتا از چند ده هزار خط کد تجاوز میکردند.

امروزه صرفا دانش در زمینه زبانهای برنامه نویسی نمی تواند پاسخگوی نیازهای نرم افزاری امروز باشد . پیچیدگی و افزایش حجم کدهای برنامه ها نیاز به همکاری و مشارکت جمع کثیری از برنامه نویسان را می طلبد. همین امر منجر به ایجاد روشها و قوانینی برای کدنویسی در نرم افزارها میگردد. امروزه یک برنامه نویس علاوه بر داشتن سواد فنی بالا نیاز به دانش در زمینه های مختلف از جمله نحوه توسعه و عملکرد توسعه برنامه ها و آشنایی با مدل های توسعه نرم افزارها دارد .

سیستم عامل لینوکس اکنون با داشتن شش میلیون خط کد یکی از بزرگترین پروژه های موفق نرم افزاری در سطح جهان است که بر روی بستر اینترنت توسعه می یابد. از این رو بررسی نحوه و مدل توسعه و برنامه نویسی در چنین سطحی برای کسی که خواهان مشارکت در این پروژه داشته لازم و ضروری بوده و همچنین مطالعه عملکرد توسعه لینوکس برای هر برنامه نویسی می تواند مفید و کارگشا باشد.

مدل توسعه سنتی برنامه :

بطور معمول برای توسعه هر برنامه ای اعمالی از جمله جمع آوری نیازها ، آنالیز و تحلیل داده های جمع آوری شده ، ایجاد مستنداتی از ویژگیهای مورد نظر و در نهایت پیاده سازی ، تست و ارتقاع کیفی

نرم افزارها صورت می گیرد . مجموعه این فعالیت ها در مدل هایی با نام متدولوژی توسعه نرم افزار قرار میگیرد که می توان از آن مدل هایی مانند آبشاری (water fall) ، نمونه سازی(prototyping) ، افزایشی (incremental) و حلزونی (spiral) و ... را نام برد.در تمامی این روشها پیگیری سلسله مراتب و رویه ای مشترک بعنوان اصلی نانوشته رعایت می گردد .

مدل توسعه هسته [1] لینوکس از کدامیک از روشها پیروی می کند؟

اولین سوالی که در مورد مدل توسعه هسته لینوکس (از این به بعد به اختصار آنرا هسته می نامیم) از کدامیک از روشهای فوق پیروی می کند ؟

در پاسخ به این سوال باید اذعان داشت که هسته دارای هیچ نوع طراحی واضحی نیست . پاسخی که آقای لینوس توروالدز به این سوال میدهد اینگونه است که :

" لینوکس صرفا یک تکامل است ونه یک طراحی هوشمندانه "

سوالی که مطرح می شود این است که چگونه این شیوه از برنامه نویسی (نداشتن متدولوژی خاص و صرفا استفاده از ایده تکاملی) در نرم افزار می تواند این پروژه بزرگ نرم افزاری را هدایت کند ؟؟؟

شروع پروژه با پیاده سازی !!!

برخلاف دیگر مدلهای متداول در توسعه نرم افزار که اغلب با اعمالی نظیر جمع آوری نیازها و مستندسازی آغاز می گردد، در توسعه هسته کدنویسی و یا همان پیاده سازی اولین مرحله در توسعه قلمداد می گردد.

بطور کلی در این شیوه توسعه نرم افزار در چهار مرحله اجرا می شود که عبارت اند از :

- --کدنویسی : در این مرحله برنامه نویس ایده و طرح اولیه خود رادر غالب کدهایی که می نویسد به هسته لینوکس اضافه می نماید (لازم به ذکر است که اضافه نمودن کد به هسته مراحل مختلفی دارد که در ادامه بحث می گردد).
- بازبینی کد : در این مرحله پس از برنامه نویسی برای بررسی صحت و کمبودهای احتمالی در کدهای نوشته شده و همچنین ترکیب حاصل از کدهای قدیمی و کد های جدید مورد بازبینی و بررسی قرار می گیرد.
- رفع خطا[2] : در این مرحله که تقریبا همزمان با مرحله قبل صورت می پزیرد خطا ها یی که در مرحله بازبینی نمایان شدند رفع نقص می گردد.

---

1. Kernel 2. Debug

- گردآوری و انتشار : در این مرحله کدهایی که جدیدا به هسته افزوده شدند با کدهای قدیمی در هسته بعد از مراحل بازبینی و رفع نقص به عنوان نسخه ای جدید از هسته گردآوری شده و پس از افزودن شماره نسخه به آن در سایت http://www.kernel.org قرار گرفته و قابل دسترس برای عموم می باشد.

لازم به ذکر است که تمامی مراحل کدنویسی ، بازبینی ، رفع خطا ، گردآوری و انتشار بصورت موازی (در هر مرحله از این فعالیت ها) اجرا می گردد؛ برای مثال در مرحله بازبینی و رفع نقص برنامه نویسان هرکدام بطور جداگانه بر روی قسمت هایی از هسته کار می کنند؛ همچنین هرکدام می توانند مرحله ای را زودتر آغاز نموده ویا زمان بیشتری به آن مرحله اختصاص دهند.

تعامل برنامه نویسان در توسعه هسته :

کدهای نوشته شده چگونه به هسته افزوده می گردند؟

هر برنامه نویس چه مسولیتی در توسعه هسته برعهده دارد ؟

چه فرد/افرادی مدیریت کلی در توسعه هسته را بر عهده دارند ؟

افراد با چه معیارهایی شرایط شرکت و همکاری در توسعه را پیدا می کنند ؟

پاسخ به این سوالات می توانند درک بهتری نسبت به نحوه تعامل و همکاری برنامه نویسان در توسعه هسته را بدهند .

در توسعه هسته هر برنامه نویسی که بتواند کارهایی از قبیل کدنویسی ، بازبینی و یا تست و رفع عیب خطا ها در هسته را انجام دهد به راحتی می تواند در توسعه هسته شرکت داشته باشد و هیچ گونه محدودیتی در این زمینه وجود ندارد. اکنون سوالی که مطرح می گردد این است که کدهای نوشته شده چگونه به هسته افزوده می گردند ؟ و هر نفر چه مسولیتی در این زمینه دارد ؟

ساختار همکاری و تعامل برنامه نویسان هسته با یکدیگر مشابه ساختار درختی می باشد ؛ بدین صورت که در توسعه دهندگان اصلی _ همانند لینوس توروالدز و اندرو مورتون و ... _ در ریشه درخت قرار می گیرند و دیگر برنامه نوسان در سطوح دیگر از این درخت همکاری قرار می گیرند .

بطور کلی هسته لینوکس از چندین قسمت تشکیل شده که نام هر قسمت را اصطلاحا زیر سیستم می گویند. برای مثال مدیریت حافظه ، شبکه ، مدیریت پردازش ها و یا مجموعه درایور ها هریک بطور جداگانه یک زیر سیستم را تشکیل می دهند .

هرکدام از این زیر سیستم ها توسط یک یا چند برنامه نویس که اصطلاحا به آنان نگهدارنده گان زیر سیستم

(system maintainer) گفته می شود نگهداری می گردد .
به همین ترتیب در هر زیر سیستم برنامه نویسانی نیز برای نگهداری ماژولها[1] و ارتقاع آنها وجود دارند ؛ این برنامه نویسان علاوه بر توسعه و ارتقاع کدهای ماژول ها وظیفه دریافت کدهای نوشته شده از دیگر برنامه نویسان را بر عهده دارند .
هربرنامه نویسی که برای ماژولی از هسته کدی می نویسدو یا اصلاحیه [2] را ایجاد می کند ، آنرا به نگهداره ماژول میدهد.
کد های نوشته شده توسط نگهدارنده فایل بازبینی و بررسی می گردد و در صورت وجود مشکل دوباره به برنامه نویس جهت رفع خطا ارسال می گردد واین پروسه آنقدر تکرار می گردد که کد جدید خالی از هر گونه خطا گردد و در مرحله بعد کد ایجاد شده از طرف نگهدارنده ماژول به نگهدارنده زیر سیستم ارسال می گردد. (لازم به ذکر است که فرایند باز بینی و رفع خطا کد می تواند بطور مشابه دوباره تکرار گردد)
کد های نهایی از طرف نگهدارندگان زیرسیستم ها به برنامه نویسانی که به آنان نگهدارندگان هسته [3] نام دارند ارسال می شود .
در طی این پروسه کدهای نوشته شده از طرف برنامه نویسان که در برگهای درخت همکاری قرار دارند قبل از رسیدن به دست نگهدارنگان هسته چندین بار توسط افراد مختلف مورد بازبینی و رفع خطا قرار می گیرند ؛ همین امر سبب افزایش کیفیت کدهای نوشته شده در هسته می گردد.
در طی این فرایند هر برنامه نویسی که به خوبی کد نویسی انجام دهد و بتواند کدهایی با کیفیت بالا تولید کند، به دلیل بازبینی های متعددی که توسط دیگر برنامه نویسان صورت می پذیرد، کارش مورد توجه قرار گرفته و می تواند جایگاه خود را نه بر اساس رابطه و شانس بلکه بر اساس لیاقت و کارایی ارتقاع دهد.
لازم به ذکر است که نرم افزار هایی که برای مدیریت و نگهداری کدها در این پروسه استفاده می گردد [4] از انواع توزیع یافته[5] آن مانند git, bitkeeper, mercurial استفاده می گردد .

---

1. Module
2. Patch
3. Core maintainer
4. این نرم افزار ها اصطلاحا سیستم های کنترل نسخه (version control system) نام دارند .
5. Distributed

مدل توسعه سنتی در مقابل مدل توسعه جدید :

در سال ۱۹۸٤ ریچارد استالمن بنیاد نرم افزار های آزاد[1] را پایه گذاری نمود. این حرکت را می توان بعنوان اولین گام در تغییر مدل توسعه نرم افزار ها نامید. قبل از این دوران برنامه نویسان به استخدام شرکت های تجاری در می آمدند و حاصل کدنویسی آنها صرفا بصورت برنامه های تجاری و کد بسته ارایه می گردید. بدلیل نبود کد برنامه ها بصورت آزاد[2] مراحل بازبینی و رفع خطا (که در مدل توسعه هسته بسیار مورد توجه قرار می گیرد) صرفا در داخل شرکت ها صورت می گرفت و همین امر منجر به کاهش کیفیت نرم افزار های تولید شده می گردید.

با انتشار کدهاو در اختیار قرار دادن آنها به دیگر کاربران می توان از توانایی کاربران نهایی[3] برای تست نرم افزارها بهره جست .

لازم به ذکر است که ایده اولیه نرم افزار آزاد بیشتر بر روی مفهوم آزادی کاربران ار منظر کپی برداری[4] ، تغییر[5] و انتشار[6] دوباره کد برنامه بوده است .

تحولی دیگر که در مدل توسعه نرم افزارها صورت پذیرفت مربوط به سال ۱۹۹۱ با آغاز توسعه هسته لینوکس توسط لینوس توروالدز صورت گرفت . در این مدل توسعه هر کس که بخواهد می تواند درتوسعه برنامه شرکت کند . برنامه نویسان و کار بران نهای هرکدام به نحوی در این پروسه شرکت داده می شوند، بطوریکه برنامه نویسان در کدنویسی و کاربران در مرحله تست و خطایابی کدها مشارکت داده می شوند.

در این مدل، توسعه نرم افزار درانحصار یک یا چند برنامه نویس ماهر باقی نمی ماند بلکه برنامه نویسان ماهر با کمک گرفتن از دیگر برنامه نویسان به توسعه کدها می پردازند.

در سال ۱۹۹۸ اریک ریموند[7] یکی دیگر از بنیان گزاران نرم افزار آزاد ایده ایی به نام "متن باز"[8] را مطرح نمود .

---

1. Free software

2. Free

3 . کاربران نهایی کاربرانی از نرم افزار هستند که در توسعه برنامه ها نقش نداشته و صرفا از نرم افزار استفاده می کنند.

4. Copy

5. Modify

6. Redistribute

7. Eric Raymond

8. Open source

این ایده بیشتر به جنبه های فنی توسعه نرم افزار های آزاد می پردازد .
اریک ریموند در مقاله ای به نام " کلیسا و بازار " به تشریح مدل جدید توسعه هسته لینوکس پرداخت که تا آن زمان این مدل بدین صورت مورد بررسی قرار نگرفته بود .
در مدل توسعه _اصطلاحا _ کلیسایی اگرچه کد های برنامه منتشر می گردد ولی توسعه برنامه توسط عده ای معدود از برنامه نویسان صورت می پذیرد .

در بین هر دو نسخه ای که از برنامه منتشر می گردد اکثر کدهای نوشته شده بصورت انحصاری در میان عده معدودی از توسعه دهندگان تولید می گردد ؛ ارتباط بین توسعه دهندگان اصلی با کاربران نهایی بسیار محدود و کم می باشد ؛ نرم افزار هایی مانند gccو emacs با این مدل توسعه ، توسعه می یابند.

در مقابل مدل توسعه کلیسایی مدل بازاری وجود دارد . در این مدل کدهای برنامه از طریق اینترنت و در میان عموم کاربران و برنامه نویسان توسعه می یابد. در این مدل بجای استفاده از توسعه متمرکز از نوع غیر متمرکز برای توسعه بهره می جوید. هر کس می تواند به نحوی در پروسه تولید نرم افزار شرکت کرده و بادیگران همکاری نماید .

لازم به ذکر است که اریک ریموند مبدع این مدل توسعه را لینوس توروالدز می نامد.

ایده ها و نگرشهایی که با رهبری نویسنده اولیه لینوکس در توسعه آن بکار رفته ، در برخی از موارد کاملا متضاد با ایده های مدل سنتی نرم افزار ها می باشد . برای ذکر چند نمونه می توان از قوانین بروکس که در سال ۱۹۷۵بیان شده نام برد :

قانون بروکس می گوید :

> با N برابر شدن تعداد برنامه نویسان کارهای انجام شده N برابر می گردد ولی پیچیدگی ، آسیب پذیری و همچنین خطا های درون سیستم با مجذور N افزایش می یابد.

در مقابل لینوس توروالدز اینگونه بیان می کند که :

> با افزایش تعداد مشاهده کنندگان کدها خطا های سیستم به سهولت برطرف می گردد.

همچنین بروکس مطرح می کند که :

> طراحی اولیه و اصلی یک نرم افزار باید از یک ذهن ویا از تعداد بسیار اندکی از توسعه دهندگان ناشی گردد.

ولی در مقابل لینوس اینگونه می گوید :

> با افزوده شدن تعداد بیشتری از تست کنندگان نرم افزار و برنامه نویسان به سیستم تقریبا هر مشکلی شناسایی خواهد شد و در نهایت راه حل از طرف یکی از همین افراد ارایه خواهد شد .

پایان چرخه طولانی ارتقاع نرم افزار :

همانطور که در مدل توسعه سنتی نرم افزار بیان شد قسمت های اصلی نرم افزار توسط عده ای معدود تولید می گردد و پس از زمانی نسبتا طولانی کد های نوشته شده در اختیار دیگر برنامه نویسان و کاربران قرار خواهد گرفت . این زمان طولانی باعث بروز مشکلاتی از جمله منسوخ شدن کدها می گردد. علاوه بر این با افزایش بازه زمانی بین انتشار نسخه های متفاوت از برنامه باعث ایجاد فاصله میان کاربران نهایی و برنامه نویسان می گردد که این امر منجر به عدم درک صحیح از نیازهای کاربران توسط برنامه نویسان می گردد.

دل مدل توسعه لینوکس به دلیل چرخه های زمانی نسبتا کوتاه بین انتشار نسخه های ارتقاع یافته فاصله میان برنامه نویسان و کاربران کمتر شده و کاربران هرچه بیشتر در تولید برنامه ها مشارکت می یابند . لینوس تورالدز در این باره می گوید :

> سریع منتشر کن ، به کررات منتشر کن و به نظرات مشتریانت گوش بده .

منابع و ماخذ :

http://www.kroah.com/log/linux/ols_2006_keynote.html

http://www.groklaw.net/article.php?story=20050529095918381

http://video.google.com/videoplay?docid=-519065200490542080

www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar

http://p2pfoundation.net/Linux_Kernel_Development_Process

http://ldn.linuxfoundation.org/how-participate-linux-community