ماژول های کرنل

امروزه در میان کامپیوتری‌ها کمتر کسی پیدا می‌شود که درباره لینوکس و کاربردهای آن چیزی نشنیده باشد. هر یک از ما به نحوی در حال استفاده از این ساخته لینوس تورووالدز هستیم، گاهی حتی بدون آنکه اطلاع داشته باشیم. کرنل لینوکس که پیوسته توسط متخصصان نرم‌افزار در حال توسعه و به‌روزرسانی است، هسته و قلب سیستم‌عامل‌های لینوکسی است. کرنل وظیفه برقراری ارتباط میان کاربر و اجزای مختلف سیستم‌عامل و سخت‌افزار سیستم را دارد، بنابراین نیاز است که امکانات بسیاری را در خود داشته باشد، اما آیا امکان اضافه کردن تمامی عملکردهای موردنیاز برای تمامی سیستم‌های جهان در کد کرنل وجود دارد؟ قاعدتا نه. اینجاست که کرنل امکان جذابی را معرفی می‌کند که اجازه می‌دهد هر کسی، بدون اینکه نیاز به تغییر کد کرنل داشته باشد، قابلیت‌های موردنیاز خود را به آن اضافه کند: ماژول‌های کرنل. در این نوشتار با این ماژول‌ها و نحوه ساخت و استفاده از آنها آشنا می‌شویم.

عرفان صابری
دانشجو مهندسی کامپیوتر
دانشکدگان فارابی دانشگاه تهران
erfansaberiow@gmail.com

کرنل لینوکس که اولین بار در سال ۱۹۹۱ توسط لینوس تورووالدز، دانشجوی ۲۱ ساله علوم کامپیوتر در دانشگاه هلسینکی فنلاند خلق و منتشر شد، هم‌اکنون توسط متخصصان و توسعه‌دهندگان بسیاری از سراسر دنیا در حال توسعه است و به طور پیوسته در حال به‌روزرسانی و بهبود است. این متخصصان دائما در حال اضافه کردن قابلیت‌های جدید به کرنل، اصلاح اشکالات و بهبود قابلیت‌های موجود و گاهی نیز حذف قابلیت‌های قدیمی و منسوخ شده از کرنل هستند. گاهی با توجه به پیشرفت‌های جدید دنیای فناوری نیاز می‌شود که پشتیبانی از سخت‌افزارها و پردازنده‌های جدید به کرنل اضافه شود، گاهی نیز نیاز است پشتیبانی از پروتکل‌های جدید به هسته افزوده شود، گاهی مشکلات امنیتی پیدا می‌شوند که نیاز به بررسی و حل دارند و همچنین موقعیت‌ها و تغییرات دیگر که باید در طول زمان ادامه پیدا کنند تا لینوکس، بتواند به خدمت‌رسانی به کاربران ادامه دهد.

از آنجا که کرنل لینوکس به شکل متن‌باز در دسترس کاربران قرار دارد، هر کسی بسته به نیاز خود می‌تواند تغییراتی در آن ایجاد کند و سپس آن را کامپایل و استفاده کند؛ اما ایجاد تغییر در کد کرنل می‌تواند منجر به این شود که دریافت به‌روزرسانی‌های جدید و پیشروی همگام با جامعه لینوکس پیچیده و سخت شود، چرا که هر تغییری که در کد اصلی کرنل لینوکس ایجاد شود، هر اشکالی که رفع شود و هر قابلیتی که حذف یا اضافه شود، باید به شکل دستی توسط افرادی که کد کرنل را برای خود شخصی‌سازی کرده‌اند به کرنل شخصی‌سازی‌شده اضافه شود. این موضوع می‌تواند باعث افزایش پیچیدگی، افزایش هزینه و زمان توسعه و در نهایت، به پایان رسیدن عمر یک پروژه نرم‌افزاری شود، بنابراین تغییر کرنل به جهت افزودن قابلیت‌های جدید ممکن است چندان راه مناسبی نباشد. پس برای افزودن قابلیت‌هایی که نیاز داریم به کرنل لینوکس چه باید کنیم؟ در اینجا، پای ماژول‌های کرنل به میان می‌آید.

کرنل‌ماژول‌ها (Kernel Modules)

از آنجا که تغییر کد کرنل و کامپایل مجدد آن می‌تواند هزینه گزافی برای افراد و سازمان‌ها داشته باشد، لینوکس قابلیتی را معرفی می‌کند به نام کرنل‌ماژول؛ کرنل‌ماژول‌ها قطعه‌هایی از کد هستند که می‌توانند در زمان اجرا به کرنل اضافه یا از آن حذف شوند. این ماژول‌ها به توسعه‌دهنده‌ها این امکان را می‌دهند که قابلیت‌های موردنیاز خود را در هر زمان که تمایل داشتند، بدون نیاز به کامپایل یا راه‌اندازی مجدد (reboot) کرنل به آن اضافه کنند.

درواقع، کرنل‌ماژول‌ها برنامه‌های کوچکی هستند که توسط افراد نوشته می‌شوند، کامپایل می‌شوند و در مواقع نیاز داخل کرنل بارگذاری و استفاده و در صورت عدم لزوم نیز، از کرنل حذف می‌شوند.

ساختار

هر ماژول از مجموعه‌ای از کدها و توابع تشکیل شده که با کرنل در ارتباط هستند و می‌توانند از توابع و سرویس‌های ارائه شده توسط کرنل استفاده کنند. این ماژول‌ها غالبا به زبان سی (C) و اسمبلی (Assembly) و به تازگی به زبان راست (Rust) نوشته می‌شوند و از بخش‌های زیر تشکیل شده‌اند:

کد منبع (Source Code): مجموعه دستورات و توابع تعریف شده توسط کاربر که به زبان‌های پشتیبانی شده توسط کرنل نوشته می‌شوند (سی، اسمبلی یا راست)

Makefile: فایلی است که شامل دستورات و تنظیمات موردنیاز برای کامپایل کد منبع است. ابزاری به نام make وجود دارد که می‌تواند بر اساس تنظیمات موجود در این فایل تنظیمات، با یک دستور، تمامی مراحل کامپایل کد را به سادگی انجام دهد.

همچنین کد منبع نیز خود شامل بخش‌های زیر است:

هدرها (header): هدرها بخش ابتدایی کدهای سی هستند که نشان می‌دهند هر برنامه قرار است از چه امکاناتی استفاده کند، به عنوان مثال توابع ریاضیاتی، ورودی و خروجی‌ها و ... . در ماژول‌های کرنل نیز از هدرهای ارائه شده توسط لینوکس مانند linux.h و module.h استفاده می‌کنیم تا ماژول موردنظر خود را ایجاد کنیم.

توابع ابتدا و خروج (init & exit functions): تابع ابتدا (init) در زمان بارگذاری شدن ماژول داخل کرنل صدا زده می‌شود و وظیفه دارد تنظیمات ابتدایی و اولیه را انجام دهد. تابع خروج (exit) نیز وظیفه دارد در زمان حذف ماژول، تمیزکاری‌های لازم را انجام دهد.

چرخه حیات

ماژول‌ها باید پس از نوشته شدن، کامپایل و سپس در سیستم بارگذاری شوند. بنابراین چرخه حیات یک ماژول را می‌توان به این شکل بیان کرد:

۱- کامپایل: ماژول باید پیش از بارگذاری در سیستم، کامپایل شود. بدین منظور از دستور make استفاده می‌کنیم که با توجه به دستورات موجود در Makefile، با کمک ابزارهای ساخت (build tools) و ابزارهای موجود در کرنل، عمل کامپایل ماژول را انجام می‌دهد و به عنوان خروجی، فایلی با پسوند .ko به ما می‌دهد که ماژول قابل بارگذاری ماست.

۲- بارگذاری: در این فاز، حیات ماژول آغاز می‌شود و تنظیمات ابتدایی آن شروع به انجام شدن می‌کنند. در ابتدا تابع ابتدا (init) صدا زده می‌شود، این تابع انجام تنظیمات اولیه و فرآیندهای آماده‌سازی را برعهده دارد و همچنین بررسی می‌کند

که آیا شرایط لازم برای بارگذاری ماژول در کرنل فراهم است یا نه، این مرحله می‌تواند شامل بررسی وجود وابستگی‌ها (dependencies)، متغیرهای محیطی (environment variables) و اختصاص منابع موردنیاز باشد.

* بارگذاری یک ماژول در سیستم‌عامل لینوکس با استفاده از دستور insmod انجام‌می‌شود.

۳- اجرا: در این فاز، ماژول می‌تواند با کرنل و دیگر ماژول‌ها ارتباط برقرار کرده و وظایف خود را انجام دهد. در این مرحله است که ماژول امکان دسترسی به منابع سیستم، ارتباط با دستگاه‌ها، بررسی سیستم و ... را دارد که به همگی این امکانات از طریق API های سیستم دسترسی پیدا می‌کند.

۴- حذف: در این مرحله دیگر نیازی به ماژول وجود ندارد و تصمیم گرفته شده که ماژول از کرنل حذف شود، بنابراین لازم است ماژول تمیزکاری‌های لازم را پیش از خروج از کرنل انجام دهد. در این فاز، تابع خروج (exit) اجرا شده و مراحل خروج شامل حذف‌فایل‌های موقتی، آزادسازی منابع، قطع اتصال از کرنل و ... را انجام می‌دهد. در این مرحله ماژول مطمئن می‌شود که پس از حذف، اثر جانبی‌ای در سیستم‌باقی‌نمی‌گذارد.

* حذف یک ماژول در لینوکس با استفاده از دستور rmmod انجام می‌شود.

ساخت و اجرای یک کرنل‌ماژول

از آنجا که کرنل‌ماژول قرار است ارتباط مستقیم با کرنل سیستم‌عامل داشته باشد، مهم است که در مراحل طراحی و ساخت آن دقت لازم به عمل بیاید، چرا که وقوع اشکال در ماژول می‌تواند منجر به اختلال در کارکرد کرنل شود. در ایجاد کرنل‌ماژول‌ها مراحل زیر دنبال می‌شوند:

۱- تهیه محیط توسعه: اولین قدم در ایجاد کرنل‌ماژول‌ها، نصب ابزارها و آماده‌سازی محیط توسعه است. در این مرحله نیاز است ابزارهایی که برای ساخت (build) ماژول و کامپایل کدها نیاز داریم را نصب کنیم. همچنین یک IDE مناسب برای خود انتخاب و ابزارهای کمکی برنامه‌نویسی سی یا اسمبلی را در آن نصب می‌کنیم. دستورات زیر، جهت نصب ابزارهای موردنیاز برای ساخت ماژول استفاده می‌شوند:

Ubuntu / Debian:

```
$ sudo apt-get install build-essential kmod
```

Arch Linux:

```
$ sudo pacman -S gcc kmod
```

همچنین جهت نصب و راه‌اندازی محیط توسعه موردنظر نیز می‌توان از میان محیط‌های موجود، موردی را با توجه به منابع سخت‌افزاری موجود، نیازها، هزینه‌ها و ... برگزید. پس از نصب محیط توسعه نیز لازم است افزونه‌های لازم بر روی آنها نصب و تنظیمات موردنظر بر روی آنها اعمال شود. پس از این موارد لازم است هدرفایل‌های موردنیاز نیز در سیستم نصب شوند تا در کد ماژول از آنها استفاده کنیم

پیش از شروع، جهت بررسی ماژول‌های موجود در کرنل می‌توان از دستور زیر استفاده کرد:

```
$ sudo lsmod
```

این دستور نشان می‌دهد که چه ماژول‌هایی در کرنل بارگذاری شده‌اند، هر کدام از آنها چه مقدار حجم دارند و نیز در حال حاضر توسط چند برنامه در حال استفاده هستند. همچنین با توجه به اینکه ماژول‌ها در دایرکتوری /proc/modules قابل مشاهده هستند، لیست ماژول‌ها را با دستور زیر نیز می‌توان دریافت نمود:

```
$ sudo cat /proc/modules
```

۲- نوشتن کد ماژول: در این مرحله، کد ماژول به یکی از زبان‌های سی، اسمبلی یا راست نوشته می‌شود. مثال زیر، یک نمونه کد منبع کرنل‌ماژول است که در زمان بارگذاری داخل کرنل پیغام Hello World و در زمان حذف از کرنل، پیغام Goodbye را ثبت می‌کند.

```c
#include <linux/module.h>
#include <linux/printk.h>

int init_module(void)
{
    pr_info("Hello world\n");
    return 0;
}

void cleanup_module(void)
{
    pr_info("Goodbye world\n");
}

MODULE_LICENSE("GPL");
```

۳- نوشتن Makefile: جهت کامپایل کردن ماژول باید از ابزار make استفاده کنیم. این ابزار تنظیمات خود را از فایلی به نام Makefile می‌خواند که باید در دایرکتوری کدمنبع ماژول قرار بگیرد. در این فایل، توضیح داده می‌شود که به ازای هر دستوری (مانند make, make clean و ...) چه مجموعه‌ای از دستورات باید اجرا شوند تا برنامه کامپایل، تست و ... شود. در زیر، نمونه‌ای از یک Makefile را می‌بینیم:

```
obj-m += hello.o

PWD := $(CURDIR)

all:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules

clean:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

۴- کامپایل کردن ماژول: برای کامپایل کدمنبع، از دستور make استفاده می‌کنیم. با اجرای این دستور، کدمنبع ما به فایل ماژول تبدیل می‌شود که توسط کرنل قابل بارگذاری و اجراست.

```
$ make
```

۵- بارگذاری ماژول: برای بارگذاری ماژول در کرنل، از دستور insmod استفاده می‌کنیم و ماژول را وارد کرنل می‌کنیم. اگر بارگذاری موفقیت‌آمیز باشد، ماژول شروع به اجرا می‌کند. این دستور به طور خودکار تابع init ماژول را صدا می‌زند.

```
$ sudo insmod hello.ko
```

می‌توانیم پیش از بارگذاری ماژول نیز اطلاعات آن را با دستور modinfo دریافت و مشاهده کنیم.

```
$ modinfo hello.ko
```

۶- بررسی لاگ‌ها: هنگامی که ماژول اجرا می‌شود، از خود لاگ‌ها و پیام‌هایی در سیستم به جای می‌گذارد. از آنجا که ماژول در کرنل اجرا می‌شود، نه در کنسول، پیام‌های آن نیز در کنسول چاپ نمی‌شوند (در اینجا دسترسی به دستورات user space مانند scanf و printf نداریم، به همین دلیل در کدمنبع ماژول از pr_info یا printk استفاده می‌کنیم). برای خواندن پیام‌ها و لاگ‌های ماژول می‌توانیم از دستورات journalctl و dmesg استفاده کنیم.

```
$ sudo journalctl –since “۱ hour ago” | grep kernel
```

```
$ sudo dmesg
```

۷- استفاده از ماژول: هم‌اکنون که ماژول در کرنل در حال اجراست، به انجام وظایف خود می‌پردازد و در صورت نیاز می‌توانیم از توابع و سرویس‌هایی که ماژول در اختیارمان قرار می‌دهد استفاده کنیم.

۸- حذف ماژول: پس از اتمام کار با ماژول می‌توانیم با استفاده از دستور rmmod آن را از کرنل حذف کنیم. این دستور به طور خودکار تابع exit را صدا می‌زند تا پاکسازی‌های لازم را انجام دهد.

```
$ sudo rmmode hello
```

استفاده از APIهای کرنل

در مرحله نوشتن برنامه، هدرفایل‌هایی را از کرنل وارد برنامه کردیم؛ این هدرفایل‌ها شامل توابعی هستند که با سرویس‌های کرنل کار می‌کنند و امکان انجام عملیات‌های مختلف را به ما می‌دهند. توابع init و exit که پیش‌تر راجع به آنها صحبت کردیم نیز از همین سرویس‌ها هستند. به جز این توابع، توابع و اشیا دیگری نیز وجود دارند که بسته به نیاز می‌توانیم از آنها استفاده کنیم:

- کار با پارامترها مانند module_param و module_param_array
- کار با حافظه مانند kmalloc و kfree
- کار با فایل‌سیستم مانند vfs_read
- ارتباط با فرآیندها مانند current و current->pid
- کار با دستگاه‌ها مانند register_chrdev
- کار با شبکه مانند sock_create و sock_sendmsg
- کار با تایمرها مانند init_timer و mod_timer

فایل‌سیستم‌های /proc و /dev در لینوکس

لینوکس همانند دیگر سیستم‌عامل‌ها از فایل‌سیستم‌های مختلفی پشتیبانی می‌کند، اما در کنار اینها، چند فایل‌سیستم مجازی (Virtual File System) نیز دارد که اجازه برقراری ارتباط میان کرنل و فایل‌سیستم‌های واقعی را می‌دهد. کرنل از این فایل‌سیستم‌های مجازی برای در اختیار قرار دادن اطلاعات فرآیندها و سخت‌افزارها بین خود و دیگر فرآیندها استفاده می‌کند.

فایل‌سیستم /proc شامل اطلاعات فرآیندهای در حال اجرا در سیستم است. لینوکس بسیاری از اطلاعات سیستم را در این فایل‌سیستم قرار می‌دهد. به عنوان مثال، /proc/modules اطلاعات ماژول‌ها را و /proc/meminfo اطلاعات مصرف حافظه را در اختیار ما می‌گذارد.

فایل‌سیستم /dev (در گذشته از /sys استفاده میشد) اطلاعات سخت‌افزارهای متصل به سیستم را در اختیار ما قرار می‌دهد. این فایل‌سیستم رابطی برای کار با سخت‌افزارهای فعال به ما می‌دهد تا بتوانیم برای دستگاه‌ها اطلاعات بفرستیم، بخوانیم و ... .

در ماژول‌های خود می‌توانیم از این فایل‌سیستم‌ها برای بررسی وضعیت سیستم و دستگاه‌های متصل به آن استفاده کنیم و اطلاعات موردنیاز خود را دریافت کنیم یا اطلاعاتی که مدنظر داریم را برای فرآیندها و یا دستگاه‌ها ارسال کنیم.

مسائل امنیتی

با توجه به اینکه کرنل لینوکس در بالاترین سطح دسترسی در سیستم اجرا می‌شود، امکان دسترسی به تقریبا هر دستگاه و عملکردی داخل سیستم را دارد، بنابراین از آنجایی که ماژول قرار است وارد کرنل شود لازم است دقت بسیاری درباره مسائل امنیتی موجود وجود داشته باشد.

آسیب‌پذیری‌های بسیاری وجود دارند که ممکن است از سوی ماژول به کرنل وارد شوند، اما در مقابل راه‌هایی نیز وجود دارند که جلوی این آسیب‌پذیری‌ها را تا حد زیادی می‌گیرند تا احتمال وقوع چنین آسیب‌هایی پایین بیاید. برخی از نکاتی که باید به آنها توجه کرد موارد زیر هستند:

آسیب‌پذیری سرریز بافر (Buffer Overflow): گاهی اوقات لازم است ماژول، ورودی‌هایی را از کاربر، شبکه و ... دریافت کند، در اینگونه موارد آسیب‌پذیری سرریز بافر بسیار محتمل است و باید در جلوگیری از آن دقت شود. هر ورودی دریافتی برنامه باید اعتبارسنجی شود و این اطمینان باید حاصل شود که ورودی مخرب وارد قسمت‌های حساس برنامه نمی‌شود.

حساسیت در مجوزها و دسترسی‌ها: یکی از راه‌های جلوگیری از وقوع حملات و آسیب‌پذیری‌ها، مدیریت سطح دسترسی هر ماژول است. در صورتی که یک ماژول امکان دسترسی به تمامی عملکردهای حیاتی سیستم را داشته باشد، طبعا احتمال آسیب‌دیدن نیز به شکل چشم‌گیری افزایش پیدا می‌کند و ماژول می‌تواند به یک حفره امنیتی در سیستم تبدیل شود. بنابراین، لازم است همواره فقط دسترسی کارکردهای لازم به ماژول داده شود و مجوزهای داده شده در حداقلی‌ترین سطح ممکن باشند.

اصالت و اعتبار ماژول: بارگذاری یک ماژول مخرب داخل سیستم می‌تواند منجر به دسترسی خراب‌کاران به سیستم و ایجاد حفره امنیتی شود، بنابراین لازم است در نصب ماژول‌ها دقت شود که تنها ماژول‌های دارای اصالت و اعتبار که صحت و سلامت آنها تایید شده وارد سیستم

شوند. یکی از راه‌های اصالت‌سنجی موجود، بررسی امضای دیجیتال است که می‌تواند اعتبار ماژول را تایید کند.

انجام تست‌ها: ورود ماژول‌های تست‌نشده به کرنل می‌تواند منجر به وقوع حوادثی شود که پیش‌بینی نشده‌اند و امکان ایجاد اختلال در عملکرد کل سیستم را دارند. در این خصوص لازم است پیش از استفاده از ماژول، تست‌ها و تحلیل‌های عملکردی و امنیتی به طور مداوم روی ماژول‌ها انجام شوند تا از عملکرد صحیح ماژول‌ها اطمینان حاصل شود.

رمزنگاری اطلاعات: با توجه به اینکه ماژول‌ها ممکن است اطلاعات حساسی را از کرنل به کاربر یا منابع دیگر ارسال کند، لازم است این اطلاعات به شکل صحیحی رمزنگاری شوند تا از دسترسی اشخاص ثالث به آنها جلوگیری شود.

کاربردها

سوالی که هم‌اکنون به وجود می‌آید، این است که با توجه به حساسیت موجود در نوشتن ماژول‌های کرنل و سخت‌بودن اشکال‌زدایی و ...، چه دلایلی ممکن است وجود داشته باشند که باعث شوند توسعه‌دهنده به جای نوشتن برنامه در محیط کاربر (user-space)، به نوشتن برنامه در محیط کرنل (kernel-space) و ماژول‌های کرنل روی بیاورد؟ دلایل مختلفی وجود دارند.

بیشترین کاربرد ماژول‌های کرنل در نوشتن درایورهای سخت‌افزارهاست. سیستم‌عامل برای اینکه امکان برقراری ارتباط کاربر با دستگاه‌هایی مانند کارت گرافیک، کارت شبکه، اسکنر و پرینتر، دستگاه‌های USB و ... را فراهم کند، نیاز دارد که درایور این دستگاه‌ها را در اختیار داشته باشد. این درایورها در لینوکس غالبا در قالب ماژول‌های کرنل ایجاد و توسعه داده می‌شوند.

گاهی نیاز داریم که فایل‌سیستم‌های اختصاصی خودمان را بنویسیم، به عنوان مثال فایل‌سیستمی نیاز داشته باشیم که به دیتابیس ما متصل شده باشد، یا فایل‌سیستمی که اطلاعات خاصی از وضعیت شبکه را در اختیار ما بگذارد، در اینگونه موارد نیز می‌توانیم از ماژول‌های کرنل استفاده کنیم.

امکان نوشتن زمان‌بندهای اختصاصی نیز در قالب ماژول‌های کرنل وجود دارد. الگوریتم‌های بسیاری برای زمان‌بندی فرآیندها در سیستم‌عامل وجود دارند و لینوکس از الگوریتم زمان‌بندی کاملا عادلانه (Completely Fair Scheduling) استفاده می‌کند. در صورتی که قصد داشته باشیم از الگوریتم‌های دیگری برای زمان‌بندی استفاده کنیم می‌توانیم الگوریتم خود را در قالب ماژول کرنل پیاده‌سازی کرده و وارد کرنل کنیم.

ماشین‌های مجازی (VMs) نیز از ابزارهایی هستند که می‌توان با کمک ماژول‌های کرنل پیاده‌سازی کرد. ماشین‌مجازی کرنل‌محور (Kernel-based Virtual Machine) یا KVM یکی از ابزارهای مجازی‌سازی متن‌باز است که بر همین اساس پیاده‌سازی شده و در سال ۲۰۰۷ نیز در نسخه ۲.۶.۲۰ کرنل منتشر شده است.

در برخی مواقع برنامه نیاز به دسترسی به منابع سطح پایین سیستم مانند وقفه‌ها (Interrupts) و یا ساختارداده‌های داخلی کرنل دارد، در اینگونه مواقع نیز می‌توان برنامه را در قالب ماژول کرنل ایجاد کرد. در مواردی نیز برنامه سربار زیادی در جابجایی میان حالت کاربر و حالت کرنل دارد که در اینگونه مواقع نیز می‌توان برنامه را به یک ماژول کرنل تبدیل کرد تا این سربار از سیستم حذف شود. با اینکار سرعت عمل برنامه در اعمال مختلف، به عنوان مثال پاسخگویی به وقفه‌ها افزایش می‌یابد و هزینه و انرژی کمتری نیز صرف این اعمال می‌شود.

با استفاده از ماژول‌های کرنل می‌توان تغییراتی در شبکه به وجود آورد و یا ابزارهایی اضافه کرد که وظیفه مدیریت و کار با شبکه را دارند. درایورهای شبکه برای مودم‌ها، اکسس‌پوینت‌ها، کارت Wi-Fi و ...، فایروال‌ها و ابزارهای فیلترینگ، ابزارهای مانیتورینگ و مدیریت شبکه و مدیریت VLAN، افزونه‌های شبکه و ... از ابزارهایی هستند که می‌توان در قالب ماژول‌های کرنل توسعه داد و استفاده کرد.

هر سیستم‌عامل به طور پیشفرض از پروتکل‌های پرکاربرد مانند TCP/IP، ARP و ... پشتیبانی می‌کند، اما در صورتی که پروتکلی داشته باشیم که پشتیبانی از آن در سیستم‌عامل وجود نداشته باشد، می‌توانیم آن را با استفاده از یک ماژول به کرنل اضافه کنیم.

توسعه BPF و eBPF نیز از راهکارهای شناخته شده در توسعه ابزارهای شبکه در لینوکس است، BPF یا Berkeley Packet Filter و eBPF یا extended BPF نوعی از ماژول‌های کرنل هستند که امکان ایجاد تغییرات در قوانین و پردازش‌های شبکه را به کاربر می‌دهند.

امنیت سیستم نیز یکی از زمینه‌هایی است که می‌توان از ماژول‌های کرنل در آن حوزه بهره گرفت. فرض کنید دیتاسنتری به هزاران کاربر مختلف سرور داده باشد و در حال میزبانی از برنامه‌ها و اطلاعات آنها باشد، نظارت بر فعالیت این سرورها یکی از وظایف مهمی است که دیتاسنتر باید انجام دهد، زیرا فعالیت مخرب توسط یکی از این سرورها می‌تواند منجر به آسیب‌پذیر شدن کل شبکه دیتاسنتر و سرورهای دیگر شود. اضافه‌کردن یک ماژول به کرنل سیستم‌عامل‌های این سرورها می‌تواند تا حد زیادی سلامت فعالیت‌های کاربران را تضمین کند، این راهکار هم‌اکنون توسط سرویس‌دهنده‌های ابری استفاده می‌شود.

به جز مواردی که پیش‌تر گفته شد، کاربردهای بسیاری را می‌توان برای کرنل‌ماژول‌ها نام برد که پیاده‌سازی هر یک توسط متخصصین و توسعه‌دهندگان می‌تواند بررسی و سنجیده شود، با توجه به حساسیت‌های موجود، شاید پیاده‌سازی این ابزارها در قالب ماژول‌های کرنل در اکثر موارد بهترین گزینه نباشد، اما در برخی موارد بهترین گزینه و یا حتی تنها گزینه ممکن است. دنیای کامپیوتر، دنیای مبادله (trade-off) است، گاهی اولویت اول ما سرعت توسعه است و هزینه توسعه و نگهداری پایین‌تر، گاهی نیز اولویت با راندمان و پرفورمنس بالای سیستم است؛ در هر مسئله‌ای با توجه به شرایط، انتظارات، محدودیت‌ها و آینده‌نگری باید تصمیمی بگیریم که مزایا و معایب خاص خود را به دنبال خواهد داشت.

منابع

William Stallings. “Operating Systems Internals and Design Principles.” (۲۰۱۸).

Peter Jay Salzman, Michael Burian, Ori Pomerantz, Bob Mottram, Jim Huang. “The Linux Kernel Module Programming Guide” (۲۰۲۳)