بسمه تعالی

فاز اول پروژه سیستم‌های عامل

نگار رحمتی 88102026

نیلوفر صالحی 88110664

در این پروژه FreeBSD kernel را به وسیله‌ی اضافه کردن چند متغیر به کرنل تغییر دادیم.

در این فاز از پروژه فایل‌های زیر را تغییر دادیم:

/usr/src/sys/sys/proc.h

/usr/src/sys/kern/kern_switch.c

متغیر "int tickets"، به struct proc در proc.h اضافه شد و متغیر "int lottery_mode" به صورت global به kern_switch.c اضافه شد. برای دسترسی به این متغیرها چهار system call به صورت زیر تعریف کردیم:

```
int setProcessTickets(int pid, int tickets);

int getProcessTickets(int pid);

int setLotteryMode(int mode);

int getLotteryMode(void);
```

هر system call به فایل syscall.c اضافه کردیم.برای اجرا کردن دستور make این فایل‌ها را compile می‌کند و یک kernel loadable module می‌سازد (syscall.ko). با دستور kldload این moduleها را load می‌کنیم.

ماژول setProcessTickets(int pid, int tickets) به این صورت عمل می‌کند که یک process خاص که با pid آن مشخص می‌شود را در نظر می‌گیرد. برای این کار از دستور pfind استفاده می‌کند. سپس متغیر tickets را که در بخش قبل تعریف کردیم برای آن مقدار دهی می‌کند.

ماژول getProcessTickets(int pid) هم به همین صورت عمل می‌کند با این تفاوت که مقدار متغیر tickets را برمی‌گرداند.

ماژول setLotteryMode(int mode) هم تقریبا به همین صورت است. این ماژول int mode را به عنوان ورودی می‌گیرد و متغیر lottery_mode که در بخش قبل تعریف کردیم را مقدار دهی می‌کند.

ماژول getLotteryMode(void) هم به همین صورت است با این تفاوت که مقدار lottery_mode را باز می‌گرداند.

تفاوت تابع‌هایی که در سطح کاربر اجرا می‌شوند با تابع‌های سطح کرنل در این است که تابع‌های سطح کاربر در "priveledge mode" اجرا نمی‌شوند. در واقع برای اینکه تابع اجرا بشود، کرنل call را trap می‌کند. وقتی تابعی از سطح کاربر فراخوانی می‌شود، system entry vector که همه‌ی system callها را در خود دارد آن را trap می‌کند. هنگام اجرا تابع به "priveledge mode" تغییر می‌کند و call اجرا می‌شود.

مثلا در printf، هنگامی که کاربر دستور printf را call می‌کند، یک system call است که در freeBSD kernel، trap می‌شود. هنگامی که printf در کرنل trap شد، خروجی استاندارد tty است در حالی که در userspace، خروجی استاندارد putty terminal است. printf در user space در حالت priveledge نیست بنابراین نمی‌تواند در خروجی چاپ کند. اول باید در kernel، trap بشود. بنابراین خروجی استاندارد بسته به اینکه در چه mode ای باشیم متفاوت است و این موضوع تفاوت printf در سطح کاربر و کرنل را توجیه می‌کند.

Syscall دارای یک load handler است که syscall را load می کند. یک تابع اصلی دارد که یک static int بر می گرداند و syscall را در آن می نویسیم. همچنین یک struct sysent که اسم syscall و تعداد آرگومان ها در آن مشخص می شود.