

Architektura komputerów 2 – projekt

Linux keyboard driver

1. Wprowadzenie

Podczas projektu będziemy pracować na otwartym oprogramowaniu Linux – który daje możliwość poznania działania systemu operacyjnego od środka i dogłębnej analizy oraz modyfikacji jądra dostarczającego zasobów dla innych procesów w systemie .

Główną częścią systemu operacyjnego Linux jest jego jądro – Linux kernel, którego zadaniami są:

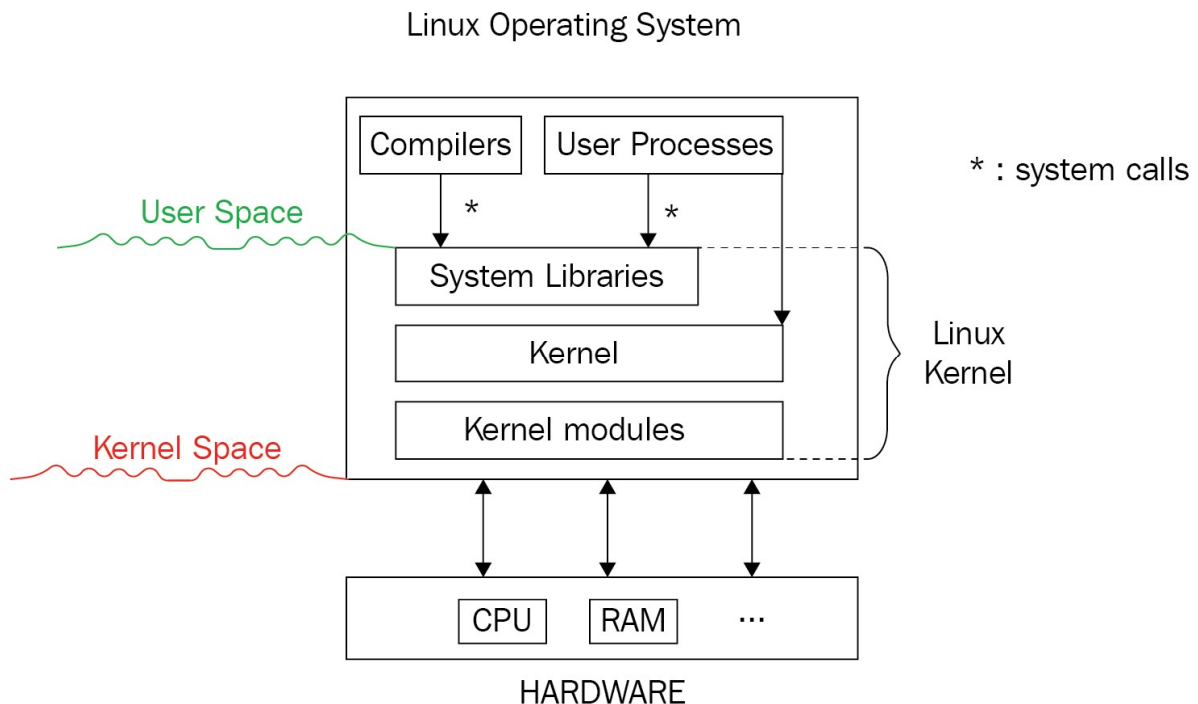
- zarządzanie pamięcią operacyjną (przydzielanie, zwalnianie oraz ochrona pamięci)
- zarządzanie procesami (tworzenie, niszczenie procesów, komunikacja międzyprocesorowa)
- **obsługa sterowników urządzeń (ładowanie i usuwanie sterowników)**
- obsługa wywołań systemowych

Kod wywoływany przez system na procesorze może zostać uruchomiony w dwóch trybach:

- Kernel mode
- User mode

Procesy wywoływane z poziomu użytkowników mają ograniczony dostęp do poszczególnych elementów hardware (CPU, pamięć), przez co szkody wyrządzone z tego poziomu nie będą tak szkodliwe jak wywołane przez w części kernela, gdzie nie ma żadnych restrykcji co do możliwości wykonywanych procesów.

1.1 Przedstawienie warstw systemu operacyjnego Linux



Większa część jądra została napisana w języku C (niektóre wstawki w assemblerze), dlatego podczas przygotowywania sterownika do obsługi klawiatury również będziemy posługiwać się językiem C.

Aby rozszerzyć podstawowe funkcje jądra można dodać do niego moduły. Dużą zaletą tej funkcjonalności jest możliwość załadowania oraz usunięcia tej części kodu, bez potrzeby restartowania całego systemu. Jednym z nich będzie nasz sterownik linux keyboard driver, który umożliwi nam obsługę podstawowej klawiatury.

Każdy moduł kernela potrzebuje posiadać co najmniej dwie funkcje:

- Funkcja załadowania modułu `init_module()`
- Funkcja usunięcia modułu `cleanup_module()`

Dodatkowo do modułu należy załadować następujące biblioteki:

- `linux/module.h`

Zapoznanie z obsługą sterowników i pracą w kernel space.

Pierwszym krokiem w celu zgłębienia tematu będzie napisanie najprostszego modułu „Hello, World”, który po załadowaniu w logach kernela wypisuje za pomocą funkcji `printk()` „Hello, World” a przy usunięciu modułu wypisuje „Goodbye, World”.

Chcąc załadować moduł, należy napisać plik Makefile, a następnie go komendą `make`. Powstały w ten sposób plik z rozszerzeniem `.ko` należy załadować do kernela komendą `insmod`. Za pomocą funkcji `lsmod` możemy zobaczyć załadowane moduły.

```
mikolaj@mikolaj-VirtualBox:~$ sudo insmod ./hello-1.ko
[sudo] password for mikolaj:
mikolaj@mikolaj-VirtualBox:~$ lsmod
Module                  Size  Used by
hello 1                 16384  0
isofs                   53248  1
vboxnetadp              28672  0
vboxnetflt              28672  0
vboxdrv                 573440  2 vboxnetadp,vboxnetflt
intel_rapl_msr          20480  0
binfmt_misc             24576  1
snd_intel8x0            49152  2
snd_ac97_codec          176128  1 snd_intel8x0
ac97_bus                 16384  1 snd_ac97_codec
intel_rapl_common       40960  1 intel_rapl_msr
snd_pcm                 155648  2 snd_intel8x0,snd_ac97_codec
crct10dif_pclmul        16384  1
snd_seq_midi            20480  0
ghash_clmulni_intel     16384  0
nls_iso8859_1           16384  1
joydev                  32768  0
snd_seq_midi_event      16384  1 snd_seq_midi
aesni_intel             376832  0
crypto_simd             16384  1 aesni_intel
cryptd                  24576  2 crypto_simd,ghash_clmulni_intel
snd_rawmidi             45056  1 snd_seq_midi
snd_seq                 77824  2 snd_seq_midi,snd_seq_midi_event
snd_seq_device          16384  3 snd_seq,snd_seq_midi,snd_rawmidi
input_leds              16384  0
snd_timer               40960  2 snd_seq,snd_pcm
snd                     114688  11 snd_seq,snd_seq_device,snd_intel8x0,snd_timer,snd_ac97_codec,snd_pcm,snd_rawmidi
serio_raw               20480  0
soundcore               16384  1 snd
vboxguest               45056  0
mac_hid                 16384  0
sch_fq_codel            24576  2
vmwgfx                  372736  2
```

Dmesg | tail -> pozwala na zobaczenie kilku ostatnich logów kernela

```
mikolaj@mikolaj-VirtualBox:~$ sudo dmesg | tail
[ 29.059913] audit: type=1107 audit(1684405515.524:67): pid=689
esktop/PolicyKit1/Authority" interface="org.freedesktop.PolicyKit1
"unconfined"
exe="/usr/bin/dbus-daemon" sauid=102 hostname=? ad
[ 29.530490] audit: type=1400 audit(1684405515.992:68): apparmor
store" requested_mask="r" denied_mask="r" fsuid=1000 ouid=0
[ 30.026621] audit: type=1400 audit(1684405516.492:69): apparmor
equested_mask="r" denied_mask="r" fsuid=1000 ouid=0
[ 44.346082] audit: type=1326 audit(1684405532.080:70): audit=100
re" sig=0 arch=c000003e syscall=93 compat=0 ip=0x7f344ee4539b code
[ 322.218191] Hello world 1.
[ 330.915534] loop24: detected capacity change from 0 to 129952
[ 386.775295] loop8: detected capacity change from 0 to 1824936
[ 386.992887] audit: type=1400 audit(1684405874.730:71): apparmor
="apparmor_parser"
[ 386.996175] audit: type=1400 audit(1684405874.734:72): apparmor
comm="apparmor_parser"
```

Na powyższym rzucie ekranu widać, że moduł po załadowaniu zgodnie z oczekiwaniami wypisał w logach kernela „Hello world 1”.

W celu usunięcia modułu należy użyć komendy `rmmod`.

```
mikolaj@mikolaj-VirtualBox:~$ sudo dmesg | tail
exe="/usr/bin/dbus-daemon" sauid=1
[ 29.530490] audit: type=1400 audit(1684405515.9
store" requested_mask="r" denied_mask="r" fsuid=10
[ 30.026621] audit: type=1400 audit(1684405516.4
equested_mask="r" denied_mask="r" fsuid=1000 ouid=
[ 44.346082] audit: type=1326 audit(1684405532.0
re" sig=0 arch=c000003e syscall=93 compat=0 ip=0x7
[ 322.218191] Hello world 1.
[ 330.915534] loop24: detected capacity change fr
[ 386.775295] loop8: detected capacity change fro
[ 386.992887] audit: type=1400 audit(1684405874.7
="apparmor_parser"
[ 386.996175] audit: type=1400 audit(1684405874.7
comm="apparmor_parser"
[ 452.026273] Goodbye world 1.
```

Po ponownym sprawdzeniu logów, możemy zauważyć, informacje o wyładowaniu modułu „Goodbye world 1”.

Kolejnym etapem w realizacji naszego projektu było napisanie character device driver.

Każdemu sterownikowi przypisany jest unikalny numer Major . Wszystkie urządzenie z tym samym numerem Major, kontrolowane są przez ten sam sterownik. Numer Minor mówi sterownikowi jakiego rodzaju jest urządzenie, które kontroluje.

Wpierw załadujemy skompilowany moduł do kernela.

```
antek@antek-VirtualBox:~$ sudo insmod dev_nr.ko
```

Wyświetlając logi w kernelu, możemy zobaczyć, że urządzenie zostało zarejestrowane.

```
[ 5598.501836] Inicjalizacja modulu  
[ 5598.501841] dev_nr - zarejsetrowano urzadzenie numer Major: 90, Minor: 0
```

Za pomocą polecenia `mknod` tworzymy urządzenie o podanych numerach.

```
antek@antek-VirtualBox:~$ sudo mknod /dev/my_device c 90 0
```

Następnie przy użyciu `ls -al` wyświetlamy pliki znajdujące się w podanej ścieżce.

```
antek@antek-VirtualBox:~$ ls /dev/my_device -al  
crw-r--r-- 1 root root 90, 0 maj 28 16:20 /dev/my_device
```

Odpalamy program test, który otwiera i zamyka nasze urządzenie.

```
antek@antek-VirtualBox:~$ ./test  
Opening was successfull!  
antek@antek-VirtualBox:~$ sudo dmesg | tail -n 2  
[ 6243.561187] dev_nr - wywolano otwarcie!  
[ 6243.561212] dev_nr - wywolano zamkniecie!
```

Chcąc sprawdzić czy nasze nowo dodane urządzenie znajduje się na liście urządzeń, sprawdzamy zawartość pliku `devices` (jak widać nasze urządzenie `my_dev_nr` jest obecne na liście).

```
antek@antek-VirtualBox:~$ cat /proc/devices  
Character devices:  
1 mem  
4 /dev/vc/0  
4 tty  
4 ttyS  
5 /dev/tty  
5 /dev/console  
5 /dev/ptmx  
5 ttyprintk  
6 lp  
7 vcs  
10 misc  
13 input  
21 sg  
29 fb  
89 i2c  
90 my_dev_nr  
99 ppdev
```

Po usunięci modułu możemy zauważyć że nasze urządzenie zniknęło z listy.

```
antek@antek-VirtualBox:~$ sudo rmmod dev_nr
antek@antek-VirtualBox:~$ cat /proc/devices
Character devices:
1 mem
4 /dev/vc/0
4 tty
4 ttyS
5 /dev/tty
5 /dev/console
5 /dev/ptmx
5 ttyprintk
6 lp
7 vcs
10 misc
13 input
21 sg
29 fb
89 i2c
99 ppdev
108 ppp
116 alsa
```

Źródła:

<https://subscription.packtpub.com>

<https://www.redhat.com/en/topics/linux/what-is-the-linux-kernel>

<https://tldp.org/LDP/lkmpg/2.6/html/lkmpg.html#AEN27>

https://github.com/Johannes4Linux/Linux_Driver_Tutorial