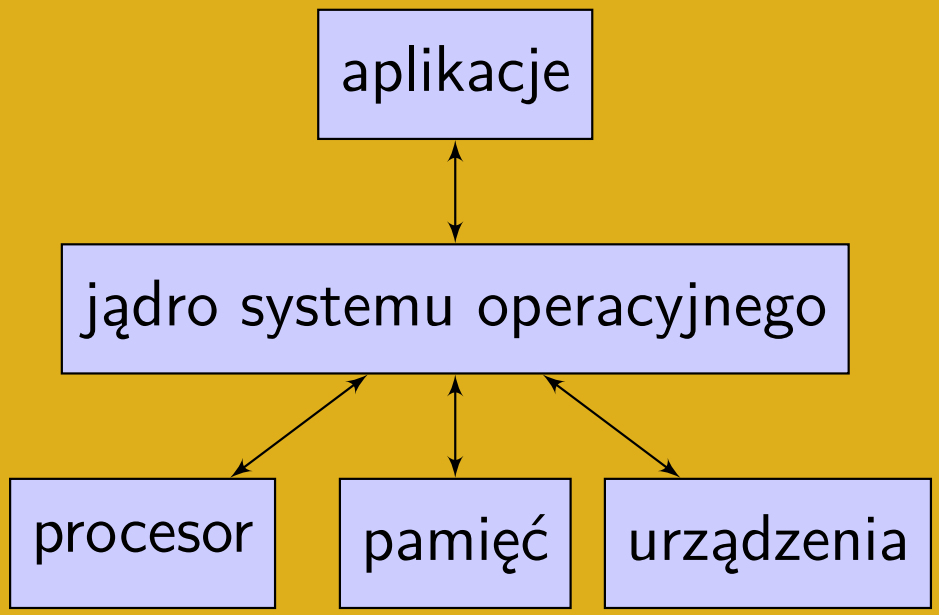
Moduły jądra

Przemysław Białkowski

12 czerwca 2013

Moduł jądra linuksa

Linuks jest *jądrem monolitycznym* z ładowalnymi modułami. Moduły zgodne z daną wersją jądra mogą być ładowane i wyładowane w trakcie jego pracy.



Moduł jądra linuksa

Moduły pracują najczęściej w trybie *Ring 0* z pełnym dostępem do pamięci, sprzętu i procesora.

Moduł jądra linuksa

Moduły pracują najczęściej w trybie *Ring 0* z pełnym dostępem do pamięci, sprzętu i procesora.

1. Brak ochrony pamięci

Moduł jądra linuksa

Moduły pracują najczęściej w trybie *Ring 0* z pełnym dostępem do pamięci, sprzętu i procesora.

1. Brak ochrony pamięci
2. Brak standardowej biblioteki C

Moduł jądra linuksa

Moduły pracują najczęściej w trybie *Ring 0* z pełnym dostępem do pamięci, sprzętu i procesora.

1. Brak ochrony pamięci
2. Brak standardowej biblioteki C
3. brak malloc

Moduł jądra linuksa

Moduły pracują najczęściej w trybie *Ring 0* z pełnym dostępem do pamięci, sprzętu i procesora.

1. Brak ochrony pamięci
2. Brak standardowej biblioteki C
3. brak malloc
4. Problem konkurencyjności

Makefile

```
ifneq ($(KERNELRELEASE),)
obj-m := hello.o

# w przeciwnym wypadku kompiluj z linii
# polecen

else
KERNELDIR ?= /lib/modules/$(shell uname -r)/
    build
     PWD := $(shell pwd)

default:
$(MAKE) -C $(KERNELDIR) M=$(PWD) modules

endif
```

hello.c

```
#include <linux/init.h>
#include <linux/module.h>

MODULE_LICENSE("Dual BSD/GPL");

static int hello_init(void) {
        printk(KERN_ALERT "Hello, world");
        return 0;
    }

static void hello_exit(void) {
    printk(KERN_ALERT "Goodbye, cruel world");
}

module_init(hello_init);
module_exit(hello_exit);
```

Parametry modułu

```
static char *whom = "world";
static int howmany = 1;
module_param(howmany, int, S_IRUGO);
module_param(whom, charp, S_IRUGO);
```

Parametry modułu

```
static char *whom = "world";
static int howmany = 1;
module_param(howmany, int, S_IRUGO);
module_param(whom, charp, S_IRUGO);
```

Można je ustawić w trakcie ładowania

```
modprobe hello howmany=10 whom="xxx"
```

Przydzielanie pamięci

```
// Pojedyncze strony
unsigned long __get_free_page(int priority);
void free_page(unsigned long addr);
// gwarantuje wyzerowanie strony
get_free_page(int priority)
// Bloki stron
unsigned long __get_free_pages(priority,order,
    dma);
void free_pages(unsigned long addr,int order);

// Specjalne obszary o~tym samym rozmiarze
kmem_cache_t *kmem_cache_create(...);
```

Przydzielanie pamięci

Przydzielanie pamięci

```
void * kmalloc(unsigned long size)
void kfree(void * addr)

void * vmalloc(unsigned long size)
void vfree(void * addr)

//-----
void * vremap(unsigned long offset,
                        unsigned long size)
```

Urządzenia blokowe

Urządzenia blokowe

```
int register_blkdev(unsigned int major, const
    char *name);
//....
```

Urządzenia blokowe

```
static void sbull_transfer(struct sbull_dev*dev,
                           unsigned long sector,
        unsigned long nsect, char *buffer, int
            write)
{
unsigned long offset=sector*KERNEL_SECTOR_SIZE;
unsigned long nbytes=nsect*KERNEL_SECTOR_SIZE;
if ((offset + nbytes) > dev->size) {
     printk (KERN_NOTICE "Beyond-end write (%ld
         %ld)\n", offset, nbytes);
     return;
}
if (write) memcpy(dev->data + offset, buffer,
    nbytes);
else memcpy(buffer, dev->data + offset, nbytes);
}
```

Na co zwrócić uwagę

- specjalne funkcje np.: `printk`
- brak debuggera
- pamięć jądra jest permanentna
- licencje

Źródła

http://aragorn.pb.bialystok.pl/~wkwedlo/OS2-10.pdf
http://www.freesoftwaremagazine.com/articles/drivers_li
Linux Device Drivers http://lwn.net/Kernel/LDD3/