Lucrarea de Laboratator 11

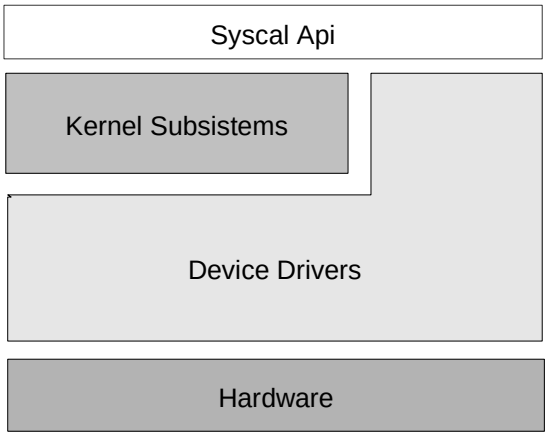
COMPILARE KERNEL / INSERARE MODULE

Curs: Utilizarea Sistemelor de Operare

Autor: Cirjan Cristian Dragos

1 . Privire de ansamblu asupra S.O. - *Kernel* (Linux – Debian / Ubuntu)

Kernel-ul este miezul unui sistem de operare, fie el Windows, Unix, Windows sau Unix based. In cazul sistemului analizat de noi, si anume Linux, kernelul este elementul ce face legatura intre functiile Api ale sistemului de operare si partea hardware a computerului.



Syscall Api este metoda traditionala a UNIX de apelare a functiilor *Kernel*-ului dintr-un *spatiu utilizator*. Implementarea sa variaza complet de la o parte la alta a *Kernel*-ului, si este complet nefunctionala pentru modulele adaugabile.

[**Source:**http://developer.apple.com/documentation/Darwin/Conceptual/KernelProgramming/boundaries/chapter_14_section_5.html]

Kernel-ul penru Linux s-a dezvoltat intotdeauna doua versiuni:

- **stable**
  recomandabila pentru a fi folosita si avand intotdeauna ca *versiune minora* un numar par;
- **unstable**
  nerecomandabila pentru a fi folosita deoarece intotdeauna apareau elemente testing si in faza experimentala destul de nocive pentru sitemul de operare, avand intotdeauna ca versiune minora un numar impar.

1.1 Denumirea versiunii de kernel

Daca tot am vorbit mai sus despre ideea de versiune minora, a venit timpul sa explic aceasta notiune. Ca exemplu, cea mai noua versiune de Kernel este: **2.6.14**. Prima secventa a versiuni se numese *versiunea majora (versiunea principala),* urmata fiind de *versiunea minora*, iar apoi de ceea ce este numit *patchlevel*. Versiunea minora, este cea care face diferenta intre doua tipuri de Kernel, si anume *stable* si *unstable*.

1.2 Scurt istoric

Primul *Kernel* pentru Linux a fost scris de *Linus Trovalds* in 1991 si imunatatatit continuu de programatori din lumea intreaga. Initial a fost contstruit pentru procesorul *Intel 80386* pentru ca apoi sa fie portat pe multe alte platforme. Intreg *Kernel*-ul a fost si este inca scris in limbajul *C*, avand cateva extensii scris sub *GNU C*[1] si *limbaj de asamblare* (sub *GNU Assembler*[2]).

Iata un mic istoric al *Kernel*-ului:
(A se mentiona faptul ca numele initial al Kernelului a fost Linux)
Septembrie 1991 – Linux 0.1
Octombrie 1991 – Linux 0.2
Decembrie 1991 – Linux 0.11 -> primul Linux self-hosted (se putea compila Linux sub Linux -> a se vedea Linux Gentoo)
Martie 1992 – Linux 0.95 -> primul Linux capabil de rulare *X Windows*[3]
Martie 1994 – Linux 1.0.0 (176.250 linii de cod)
Martie 1995 – Linux 1.2.0 (310.950 linii de cod)
Iunie 1996 – Linux 2.0.0 (777.956 linii de cod)
Ianuarie 1999 – Linux 2.2.0 (1.800.847 linii de cod)
Ianuarie 2001 – Linux 2.4.0 (3.377.902 linii de cod)
Decembrie 2003 – Linux 2.6.0 (5.929.913 linii de cod)
Octombrie 2005, 16 – Linux 2.6.14 – ultima versiune stabila

1 GNU Compiler Collection – set de compilatoare produs de GNU
2 Compilatorul GNU pentru limbaj de asamblare
3 Serverul Grafic Linux

2. Compilarea *Kernel*-ului

2.1 Pro & Contra *Kernel* compilat

Kernel-ul ppoate fi instalat in doua variante:

1. se scoate o versiune precompilata pentru distributia de Linux pe care o avem si se instaleaza
2. se scoate versiunea *vanilla*, versiune pe care o gasim pe *http://kernel.org* sau se scot sursele de kernel specifice distributiei pe care o avem si se compileaza.

Ce inseamna versiune *vanilla*? Aceasta versiune este versiunea de *Kernel* neprelucrata absolut deloc de distributiile de Linux existente la ora actuala. Nu este recomandata aceasta versiune de *Kernel* datorita posibilelor incompatibilitati ce pot aparea la nivel de distributie sistem de operare. Astfel, fiecare distributie de Linux preia ultima versiune *vanilla* si o prelucreaza pentru sine, pentru ca apoi a o lansa pentru utilizatorii sai. Acest lucru poate dura de obicei destul de mult, avand ca rezultat un foarte mare delay intre ultima versiune de *Kernel vanilla* si ultima versiune de *Kernel* aparuta pentru distributia de Linux folosita.

De ce ar fi mai utila compilarea unui *Kernel* decat folosirea versiunii precompilate? Ei bine se pot insira destul de multe motive, insa cele mai importante ar fi:

- distributiile standard precompilate de kernel pot fi inadecvate deoarece pot aparea incompatibilitati hardware sau chiar lipsa suportului pentru anumite componente
- pentru distr. precomp., prea mult suport harware activat in interiorul kernelului poate duce la:
  - folosirea a prea multa memorie de catre acesta, sau
  - lansarea sistemului de operare intr-un timp mai lung decat cel optim
- in cazul compilarii *kernel*-ului putem alege noi elemente neactivate in *kernel*-ul precompilat sau deselecta elemente de care nu avem nevoie
- putem imbunatati performantele computerului prin setarea modulelor *kernel*-ului in asa fel incat sa se potriveasca intr-un mod cat mai bun cu sistemul hardware detinut.

2.2 Compilarea Kernelului

2.2.1 Donlowad/install Kernel sources

Vom incepe prin a downloada/instala sursele pentru *Kernel* (voi scrie cu italic si pasii pentru versiunea vanilla):

```
    # apt-get install kernel-source-2.6.8
    # cd /usr/src
    # tar -xjvf kernel-source-2.6.8.tar.bz2
    # cd kernel-source-2.6.8
```

```
    # cd /usr/src
    # wget http://www.kernel.org/pub/linux/kernel/v2.6/linux-
2.6.12.2.tar.bz2
    # tar -jxvf linux-2.6.12.tar.bz2
    # cd linux-2.6.12
```

APLICATII:

1. Instalati pachetul *kernel-soruce-2.6.8* .

2.2.2 Config file (I) [Kernel update]

Si in cazul unui update de *Kernel* si in cazul unei compilari noi vom fi nevoiti sa creem fisierul de configurare pentru compilarea acestuia. Avantajul in cazul unui update este acela ca putem folosi fisierul de configurare al *Kernel*-ului vechi. De aceea vom copia acest fisier astfel (a se observa ca nu exista nici o

diferenta intre versiunile de kernel folosite, singura fiind numele directorului) in directorul surselor de *Kernel* pe care dorim sa il instalam, sub numele de *.config*:

```
    # cd /boot
// vom cauta fisierul de configurare
    # ls
// vom copia fisierul de configurare intr-un loc sigur (spre exemplu
/home/student)
    # cp config-2.4.17-1-386 /usr/src/linux-source-2.6.12/.config
```

```
    # cd /boot
// vom cauta fisierul de configurare
    # ls
//  vom  copia  fisierul  de  configurare  intr-un  loc  sigur  (spre  exemplu
/home/student)
    # cp config-2.4.17-1-386 /usr/src/linux-2.6.12/.config
```

Urmatorul pas este acela de actualizare a fisierului de configurare cu nou Kernel, iar lucrul acesta se face cu ajutoul comenzii **make oldconfig** executata in directorul surselor noului Kernel.

```
    # make oldconfig
```

In cazul in care dorim sa folosim fisierul de cofigurare a surselor, vechi, este recomandabil ca dupa **make oldconfig** sa trecem si prin pasul doi, adica sa folosim una din comenzile de la 2.2.3 (**make config / make menuconfig / make xconfig**).

APLICATII:

1. Executati comanda **make mrproper**. (Comanda este explicata in 2.2.3).
2. Copiati fisierul de configurare al Kernel-ului vechi in directorul de se surse al noului kernel sub numele **.conf ig** .
3. Executati comanda **make oldconfig.**

2.2.3 Config file II [Kernel compile]

Configurarea surselor de Kernel se poate face in mai multe moduri:
Pas optional:

- ***make mrproper***
  [make clean configuration | practic va sterge orice fisier de configurare sau fisiere obiect pe care Kernelul vechi le avea]

```
    # make mrproper
```

Pasul de configurare propriuzisa:
Sursele de Kernel pot fi configurate cu ajutorul mai multor comenzi, insa toate fac acelasi lucru:

- **make config**
  [configurarea se face in urma unui set de intrebari despre sistem la care utilizatorul trebuie sa raspunda]

```
    # make config
```

- **make menuconfig**
  [configurarea se face printr-un meniu text, in care utilizatorul poate selecta elementele din Kernel pe care le doreste active]

```
    # make menucofig
```

- **make xconfig / make gconfig**
  [acelasi mod de configurare ca **menuconfig**, insa configurarea se face sub serverul X]

```
# make xconfig
```

- **make defconfig**
  [va folosi fisierul de configurare default]

```
# make gconfig
```

APLICATII:
1. Rulati **make menuconfig** si dezactivati optiunea pentru componentele *Ethernet 1000Mb*.

2.2.3 Componente

Majoritatea componentelo Kernel-ului ce se doresc compilate poate fi compilate in doua moduri:
- ca parte integrata in Kernel (se incarca in memorie de la bun inceput, adica de la pornirea S.O. - este ocupata mai multa memorie, insa metoda este mai viabila dpv al rapiditatii sistemului). Aceasta parte, dupa compilarea chernelului va constituie partea principala, regasita in *bzImage*
- ca modul (este o componenta de sine statatoare, incarcata de Kernel, doar in cazul in care este nevoie de ea – este ocupata mai putina memore, insa la timpul de executie este mai mic, deoarece este nevoie de incarcarea modulului). Modulele compilate vor fi adaugate in directorul */lib/modules*.

Exista si componente ce au restrictii asupra lor si nu pot fi compilate decat ca parte integrata in Kernel sau doar ca modul.

2.2.4 Utilitare

Pentru a putea configura sursele de Kernel avem nevoie de informatii despre partea hardware a sistemului nostru. In aceasta situatie ne vor fi utile urmatoarele utilitare:
- lsmod
- lspci

lsmod este o aplicatie care arata formateaza continutul */proc/modules*, prezentand ce module de kernel sunt incarcate la momentul accesarii comenzii

```
    # lsmod
Module                  Size  Used by    Not tainted
wcfxs                  28512   3
wcfxo                   8512   2
zaptel                179840  18  [wcfxs wcfxo]
usbcore                58400   1
i810_rng                2656   0  (unused)
hisax                 449444   0  (unused)
isdn                  117184   0  [hisax]
slhc                    5040   0  [isdn]
hdlc                   13592   0  [zaptel]
isa-pnp                30724   0  [hisax]
tulip                  40928   1
crc32                   2880   0  [tulip]
ide-scsi                9424   0
agpgart                39576   0  (unused)
    #
```

lspci este un utilitar ce afiseaza informatii despre toate device-urile sistemului si despre componentele conectate la acestea

```
    # lspci
0000:00:00.0 Host bridge: nVidia Corporation nForce2 AGP (different version?) (rev c1)
0000:00:00.1 RAM memory: nVidia Corporation nForce2 Memory Controller 1 (rev c1)
0000:00:00.2 RAM memory: nVidia Corporation nForce2 Memory Controller 4 (rev c1)
0000:00:00.3 RAM memory: nVidia Corporation nForce2 Memory Controller 3 (rev c1)
0000:00:00.4 RAM memory: nVidia Corporation nForce2 Memory Controller 2 (rev c1)
0000:00:00.5 RAM memory: nVidia Corporation nForce2 Memory Controller 5 (rev c1)
0000:00:01.0 ISA bridge: nVidia Corporation nForce2 ISA Bridge (rev a4)
0000:00:01.1 SMBus: nVidia Corporation nForce2 SMBus (MCP) (rev a2)
0000:00:02.0 USB Controller: nVidia Corporation nForce2 USB Controller (rev a4)
0000:00:02.1 USB Controller: nVidia Corporation nForce2 USB Controller (rev a4)
0000:00:02.2 USB Controller: nVidia Corporation nForce2 USB Controller (rev a4)
0000:00:04.0 Ethernet controller: nVidia Corporation nForce2 Ethernet Controller (rev a1)
0000:00:05.0 Multimedia audio controller: nVidia Corporation nForce MultiMedia audio [Via
VT82C686B] (rev a2)
0000:00:04.0 Ethernet controller: nVidia Corporation nForce2 Ethernet Controller (rev a1)
0000:00:05.0 Multimedia audio controller: nVidia Corporation nForce MultiMedia audio [Via
VT82C686B] (rev a2)
0000:00:06.0 Multimedia audio controller: nVidia Corporation nForce2 AC97 Audio Controler (MCP)
(rev a1)
0000:00:08.0 PCI bridge: nVidia Corporation nForce2 External PCI Bridge (rev a3)
0000:00:09.0 IDE interface: nVidia Corporation nForce2 IDE (rev a2)
0000:00:0c.0 PCI bridge: nVidia Corporation nForce2 PCI Bridge (rev a3)
0000:00:0d.0 FireWire (IEEE 1394): nVidia Corporation nForce2 FireWire (IEEE 1394) Controller
(rev a3)
0000:00:1e.0 PCI bridge: nVidia Corporation nForce2 AGP (rev c1)
0000:01:0a.0 SCSI storage controller: Adaptec AIC-7892B U160/m (rev 02)
0000:01:0b.0 RAID bus controller: Silicon Image, Inc. (formerly CMD Technology Inc) SiI 3112
[SATALink/SATARaid] Serial ATA Controller (rev 02)
0000:02:01.0 Ethernet controller: 3Com Corporation 3C920B-EMB Integrated Fast Ethernet
Controller [Tornado] (rev 40)
0000:03:00.0 VGA compatible controller: Matrox Graphics, Inc. MGA G550 AGP (rev 01)
    #
```

APLICATII:

1. Extrageti din rezultatul comenzii **lspci** tipul placii de retea.

2.2.5 Module

Asa cum am spus mai sus, anumite parti din Kernel pot fi compilate ca module. Aceste module sunt apelate si incarcate automat de catre Kernel cand este nevoie de ele. Se recomanda compilarea ca module a componentelor de Kernel ce nu sunt folosite de acesta direct de la bootare.

Ca avantaje sunt cele enumerate mai sus:

- nu consuma memorie decat in cazul in care este nevoie de ele
- sitemul (S.O.) porneste mai repede

Dezavantaje:

- incarcarea modulului necesita timp

2.2.6 Compilarea propriuzisa

Compilarea propriuzisa are trei pasi:

- **make** – contruieste imaginea de kernel
- **make modules_install** – instaleaza modulele
- copierea imaginii de Kernel, in directorul in care acesta booteaza in mod normal (de regula acest director este */boot*)

APLICATII:

1. Rulati comenzile de compilare a surselor de kernel. (Numai in cazul in care exista 30 min disponibile – cam atat ar lua MAXIM o compilare de kernel)

2.2.7 Modificarea BootLoader-ului (GRUB)

```
## Imaginea de boot initiala
title           Debian, kernel 2.6.2-5-386
root            (hd0,1)
kernel          /boot/vmlinuz-2.6.2-5-386 root=/dev/hda2 ro quiet splash
initrd          /boot/initrd.img-2.6.2-5-386
## Imaginea de boot initiala
title           Debian, kernel 2.6.8-5-386
root            (hd0,1)
kernel          /boot/vmlinuz-2.6.8-5-386 root=/dev/hda2 ro quiet splash
initrd          /boot/initrd.img-2.8.10-5-386
```

2.2.8 Sistem reboot

Rebootarea sistemului se poate face cu ajutoul comezilor: **reboot**, **shutdown -r now**, **init 6**, **Crtl+Alt+Del**.

La repornire se alege optiunea de bootare cu noul kernel.

Pentru a putea vedea erorile de la bootare se pot folosi comenzile:

- **dmesg**
- **less /var/log/messages**

Recomandat este sa verificam si logul de creare a kernelului:

- **less ~/kernelbuild.log**

3. Incremental patch

Patch-urile incrementale fac posibila trecerea de la o versiune a *Kernel*-ului la versiunea imediat urmatoare (de exemplu de la 2.6.9 la 2.6.10). Acestea economisesc timp, iar cantitatea de data download-ata este mai mica. Odata cu aplicarea patch-ului, se reiau comenzile: **make oldconfig** si comenzile de instalare: **make**, **make modules_install**.

4. Module

Ma reintorc la module pentru a pezenta posibilitatea de a "jongla" cu modulele *Kernel*-ului chiar in timp ce acesta este activ. Astfel:

- **rmmod** – aplicatie cu ajutorul careia se pot scoate module din *Kernel* (recomandata este folosirea comenzii **modprobe -r**)
- **modprobe** – aplicatie "inteligenta" ce adauga sau scoate module din *Kernel*-ul de Linux.

```
    # modprobe -t net
// Load one of the modules that are stored in the directory tagged "net". Each module
are tried until one succeeds.
    # modprobe -a -t boot
// All modules that are stored in directories tagged "boot" will be loaded.
    # modprobe slip
// This will attempt to load the module slhc.o if it was not previously loaded, since
the slip module needs the functionality in the slhc module. This dependency will be
described in the file modules.dep that was created automatically by depmod.
    # modprobe -r slip
// This will unload the slip module. It will also unload the slhc module
automatically, unless it is used by some other module as well (e.g. ppp)
```

- **insmod** – aplicatie ce insereaza module in Kernel-ul.