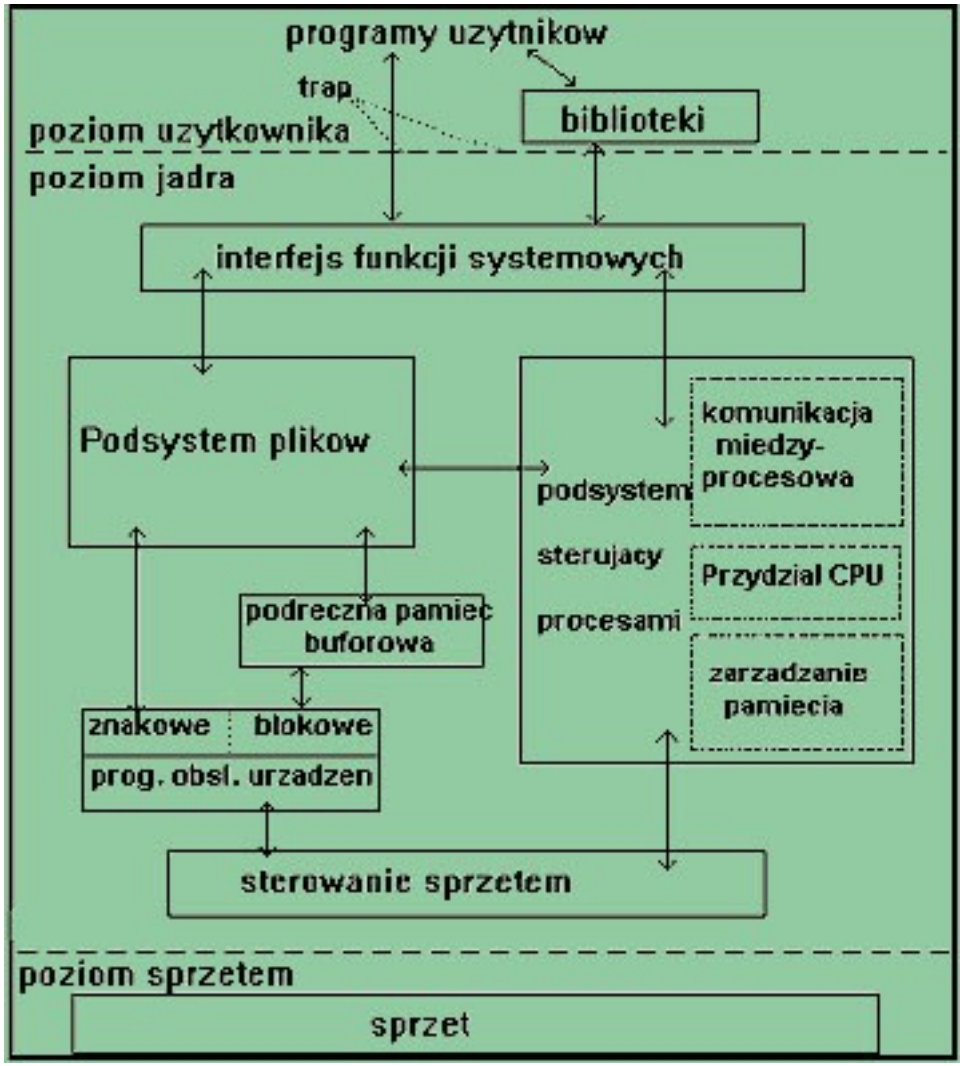
"Budowa jądra 2.6/3.x"

Krzysztof Chomski
Krystian Hanek

Jądro systemu linux

Sercem Linuxa jest jego jądro.

W jądrze Linuks sterowniki do urządzeń mogą zarówno być wbudowane w samo jądro (wkompilowane moduły), albo też w osobnym pliku, i wtedy są one ładowane do pamięci komputera i uruchamiane w razie potrzeby, lub przy starcie systemu.



Rys. 3.1. Schemat ogólny budowy jądra Linuxa

Zalety modułowej budowy

Kernel Linuks jest **makrokernelem**, tzn. zajmuje się nie tylko komunikacją pomiędzy poszczególnymi procesami, ale także zarządzaniem pamięcią, obsługą urządzeń czy obsługą systemów plików. Dla porównania, w architekturze *mikrokernela*, sam kernel zajmuje się tylko zapewnieniem komunikacji pomiędzy procesami, natomiast do obsługi poszczególnych urządzeń, czy systemów plików, zajmują się osobne wykonywalne programy.

Jednak Linuks ma modułową budowę, co oznacza że części kodu odpowiadające za obsługę poszczególnych urządzeń, systemów plików czy protokołów, mogą zostać od niego oddzielone - i dzięki temu zyskuje największe zalety mikrokerneli, czyli:

- obsługę poszczególnych urządzeń czy protokołów można rozwijać niezależnie od samego kernela (nie wymaga to zmian w innych częściach kernela)
- podział prac nad jądrem na grupy zajmujące się poszczególnymi częściami znacznie ułatwia i przyspiesza rozwój
- możliwość zmniejszenia wielkości kernela, poprzez wykasowanie z pamięci operacyjnej nieużywanych modułów
- możliwość tworzenia zamkniętych, binarnych sterowników

...Zalety modułowej budowy

Przy tym budowa jako makrokernel, i możliwość wbudowania modułów w strukturę samego kernela, przyspiesza działanie, ponieważ nie ma konieczności wielokrotnego powtarzania przełączania procesów, przy komunikacji pomiędzy poszczególnymi modułami.

W jądrze można wyróżnić dwa główne składniki:

- podsystem plików
- podsystem sterujący procesami.

Podsystem plików zarządza plikami, a podsystem sterujący procesami odpowiedzialny jest za **synchronizację procesów**, **komunikację** miedzy nimi, **zarządzanie pamięcią** i **szeregowanie procesów**.

Funkcje systemowe

Funkcje systemowe tworzą interfejs miedzy wykonywanym programem, a systemem operacyjnym.

Funkcje systemowe można z grubsza podzielić na pięć podstawowych kategorii:

- *nadzorowanie procesów*,
- *operacje na plikach*,
- *operacje na urządzeniach*,
- *komunikacja*,
- *utrzymywania informacji*.

Rodzaj funkcji systemowych

1.Nadzorowanie procesów

- zakończenie(exit),zaniechanie,
- załadowanie, wykonanie(exec),
- utworzenia(fork), zakończenie procesu(exit),
- pobranie atrybutów procesu(getegid,getgid,getpid ..)określenie atrybutów procesu (setgid,setuid),
- czekanie czasowo(wait),
- oczekiwanie na zdarzenie(wait),sygnalizacja zdarzenia(signal),
- przydział i zwolnienie pamięci.

2.Operacje na plikach

- utworzenie pliku(create);,usuwanie,
- otwarcie(open),zamkniecie(close),
- czytanie(read),pisanie(write),zmiana położenia,
- pobranie atrybutów pliku, określenie atrybutów pliku.

...Rodzaj funkcji systemowych

3.Operacje na urządzeniach

- zamówienie urządzenia, zwolnienia urządzenia,
- czytanie urządzenia, pisanie do urządzenia,
- pobranie atrybutów urządzenia, określenie atrybutów urządzenia,
- logiczne przyłączanie lub odłączenie urządzeń.

4.Utrzymywania informacji

- pobranie czasu(time,stime), określenie czasu(time),
- pobranie danych systemowych, określenie danych systemowych,
- pobranie lub określenie atrybutów procesu.

...Rodzaj funkcji systemowych

5.Komunikacja

- utworzenie, usunięcie połączenia komunikacyjnego,
- nadawanie, odbieranie komunikatów,
- przekazanie informacja o stanie,
- przyłączenie lub odłączenie urządzeń wymiennych.

Podsystem plików

Podsystem plików zarządza plikami:

- przydziela na nie pamięć,
- administruje pamięcią wolna,
- steruje dostępem do plików,
- udostępnia użytkownikom dane.

Podsystem sterujący procesami

odpowiedzialny jest za:

synchronizację procesów,

komunikację miedzy nimi,

zarządzanie pamięcią,

szeregowanie procesów.

Podsystem sterujący procesami

Podsystem sterowania procesami współpracuje z podsystemem plików podczas ładowania pliku do pamięci w celu wykonania. Podsystem sterowania procesami czyta pliki wykonywalne do pamięci przed rozpoczęciem ich wykonywania. Wtedy współpracuje z podsystemem plików . Proces jest wykonaniem programu i składa się ze zbiorowości bajtów, które CPU interpretuje jako instrukcje maszynowe, dane i stos. Jądro steruje wykonaniem procesów. Kilka procesów może być wcieleniem jednego programu. Jądro identyfikuje każdy proces za pomocą jego numeru, zwanego identyfikatorem procesu(PID).Proces 0 jest procesem specjalnym, tworzonym "ręcznie".

POBIERANIE JĄDRA

Sprawdzenie wersji aktualnego jądra:

 # uname -r

Ze strony http://www.kernel.org pobrać odpowiednie archiwum. W naszym przypadku będzie to wersja 3.9.4:

 # wget https://www.kernel.org/pub/linux/kernel/v3.x/linux-3.9.4.tar.xz

Po pobraniu kopiujemy/przenosimy archiwum do **/usr/src** i rozpakowujemy:

 # cp linux-3.9.4.tar.xz /usr/src/

 # tar -xJf linux-3.9.4.tar.xz

Następnie tworzymy dowiązanie symboliczne:

 # ln -s linux-3.9.4 linux

Potrzebne pakiety do dalszej części:

 # apt-get install libncurses5-dev make gcc bin86 libc6-dev kernel-package glibc-devel egcs

Zamiast pakietu **libncurses5-dev** możemy zainstalować ***tk9.5*** dzięki czemu będziemy mogli skorzystać z XWindowsowego okna przy konfiguracji jądra.

Konfiguracja

Do kompilacji jądra systemu konieczne są zainstalowane powyższe pakiety. Przed kompilacją należy to sprawdzić.

Mamy do wyboru trzy sposoby ustawienia opcji konfiguracyjnych:

- program konfiguracyjny uruchamiany poleceniem ***make config***. Wybór opcji kompilacyjnych polega na żmudnym odpowiadaniu na zadawane przez komputer pytania. Niezbyt wygodny w użyciu, nie zalecane.

- program konfiguracyjny uruchamiany poleceniem make ***menuconfig***. Dostajemy narzędzie a`la mc [Midnight Commander]. Cały proces przygotowania pliku konfiguracyjnego opiera się na używaniu dwóch klawiszy: entera i spacji. Bardzo prosty w obsłudze, za pomocą znaku "?" dostajemy dość obszerną pomoc dla każdego z komponentów.

- program konfiguracyjny uruchamiany poleceniem make ***xconfig***. Jak nazwa wskazuje odpalamy go w środowisku XWindow, ma przyjemny interfejs, bardzo łatwo i fajnie się go obsługuje. Wszelkie wybory dokonujemy za pomocą myszki.

...Konfiguracja

Sterowniki mogą być wkompilowane w jądro, mogą być skompilowane jako moduł lub nie kompilowane wcale. My użyjemy do konfiguracji jądra programu ***menuconfig***. Sterowniki kompilowane na stale w jądro oznaczone będą *****, jako moduł ***M***, lub nie kompilowane wcale - ***okienko puste***.

Wybór poszczególnych rodzajów odbywa się poprzez wielokrotne naciśniecie spacji.

Konfigurację jądra zaczynamy od wydania polecenia:

 # make menuconfig

Czynimy to z poziomu **/usr/src/linux**. Uważam, iż narzucony przez nas sposób jest lepszy od wspomnianych powyżej. Dysponuje on bardzo wygodnym i przejrzystym interfejsem. Wymaga jednak zainstalowanych bibliotek ***ncurses***. Przedstawiamy krótki opis większości parametrów konfiguracyjnych, zwracając większą uwagę na te najbardziej istotne.

Opcje możliwe do zaznaczenia:

< M > - sterownik będzie skompilowany do postaci modułu

< * > - sterownik będzie wkompilowany w jądro na stałe, choć może być modułem

[*] - sterownik będzie wkompilowany w jądro, nie może być modułem

[ ], < > - sterownik nie będzie zainstalowany

Opcje konfiguracyjne jądra

ze względu na ilość opcji przedstawimy tylko parę

Code maturity level options
[ ] Prompt for developmnet and/or incomlete code/drivers
Włącza niektóre narzędzia obsługiwane przez linuksa, np. dyski sieciowe, systemy plików czy protokoły sieciowe, znajdujące się w fazie rozwojowej. Nie zaznaczajmy tej, jeśli nie jest to konieczne. Może to bowiem pogorszyć stabilność systemu.

Loadable module support
[*] Enable loadable module support
[*] Set version information on all module symbols
[*] Kernel module loader
Załączmy te opcje, w przeciwnym razie nie załadujemy żadnych modułów!

Processor type and features
(Athlon/Duron/K7) Processor family
W tym polu wybieramy typ naszego procesora

...Opcje konfiguracyjne jądra

(Off) High Memory Support
Wybieramy Off, jeśli mamy mniej niż 1GB całkowitej pamięci fizycznej

[ ] Symetric multi-processing support
Jeżeli mamy co najmniej dwa procesory, to zaznaczamy tą opcję. Jeśli mamy tylko jeden procesor, to nie zaznaczajmy jej, gdyż spowolni ona start jądra.

General setup

[ ] Support for hot-plugabble devices
Możemy załączyć nowe urządzenia na uruchomionym systemie. Warto zaznaczyć tę opcję

[*] Power Management Support
Ważne w laptopach, gdyż pozwala na oszczędzanie energii. Gdy jakieś urządzenie nie jest zajęte, to system je tymczasowo wyłącza lub "usypia".

< * > Paraller port support
Zaznaczmy tę opcję, gdyż pozwala ona na obsługę portu równoległego (np. może go wykorzystywać drukarka)

...Opcje konfiguracyjne jądra

< > RAID support
Ten sterownik pozwala łączyć kilka partycji dyskowych w jedno logiczne urządzenie blokowe.

Dostępne są poniższe sposoby:
< > Linear (append) mode
< > RAID - 0 (striping) mode
< > RAID - 1 (mirroring) mode
< > RAID-4/RAID-5 mode
Jeżeli więc wiemy, jakie macierze dyskowe chcemy utworzyć, to możemy zaznaczyć odpowiednie opcje powyżej

Cała lista opracowana: https://www.dropbox.com/s/37dgnood6igtqhl/opcje_kernel.pdf

KOMPILACJA

Teraz przyszła pora, aby skompilować nowe jądro:

make dep zrobi wszystkie powiązania, musi być podane z głównego katalogu źródeł jądra /usrc/src/linux,

make clean wyczyści niepotrzebne śmieci ze źródeł, ponieważ stare pliki pośrednie mogą wciąż istnieć po zmodyfikowaniu konfiguracji,

make zImage skompiluje nam obraz naszego jądra,

make bzImage należy użyć tego polecenia w przypadku wystąpienia błędu o zbyt dużym rozmiarze jądra,

make modules skompiluje moduły jądra,

make modules_install zainstaluje skompilowane moduły.

...KOMPILACJA

Żeby skompilować używamy kolejno:

 # make

 # make modules

 # make modules_install

 # make install

Operacja kompilacji trwa zależnie od szybkości naszego procesora oraz od ilości zainstalowanej pamięci RAM.

...KOMPILACJA

Mamy już skompilowane jądro, które leży sobie w pliku ***/usr/src/linux/arch/i386/boot/zImage*** lub jeśli użyliśmy polecenia ***bzImage*** w pliku ***/usr/src/linux/arch/i386/boot/bzImag.***

Ważne, aby przed wydaniem polecenia ***make modules_install*** usunąć z katalogu ***/lib/modules/numer-wersji*** poprzednio skompilowane moduły. Czynność tą musimy wykonać tylko w wypadku, gdy kompilujemy jądro o numerze wersji takim samym jak skompilowane wcześniej. Przykładowo jeśli mamy już jądro ***2.2.13*** i kompilujemy jądro o takiej samej wersji usuwamy katalog ***/lib/modules/2.2.13***. Uchroni nas to przed nadpisywaniem modułów oraz komunikatem ***Unresolved symbol ...***

Instalacja jądra

Jądro w Linuxie powinno zostać umieszczone w katalogu /boot i nie powinniśmy nadpisywać starego jądra, gdyż może się zdarzyć, że nowe jądro wcale nie będzie chciało z nami współpracować i wtedy zostalibyśmy na lodzie. Przekopiujemy plik jądra do katalogu /boot, nadając mu nazwę nowe-jadro.

Jądro skompresowane jest generowane domyślnie z dołączoną do niego procedurą rozpakowującą je podczas uruchamiania systemu. Ta cecha Linuxa umożliwia umieszczenie minimalnego, lecz kompletnego, systemu obsługującego cały sprzęt na jednej dyskietce. Czas tracony na dekompresję jest znikomy.

KONFIGURACJA GRUB

1. Otwieramy plik ***menu.lst*** zawierający listę dostępnych systemów operacyjnych:

```
# cd /boot/grub/
# nano menu.lst
```

2. Dodajemy na końcu pliku nową sekcje:

```
title Debian GNU/Linux, kernel 3.9.4
root (hdX,X)
kernel /boot/vmlinuz-3.9.4 root=/dev/hdaX
ro vga=792
initrd /boot/initrd.img-3.9.4
savedefault
```

3. Zapisujemy plik

Gdzie (hdX,X) podajemy wartości z poprzednich

wpisów. Tak samo postępujemy z root=/dev/hdaX.

Błędy "Kernel Panic"

Błędy "**Kernel Panic**", występują w czasie ładowania się systemu, są one wynikiem błędnej kompilacji lub konfiguracji jądra.

VFS: Cannot open root device (0:0):

Numer może być przypadkowy, jest to nazwa urządzenia. Błąd ten występuje, jeśli zapomnimy podać w jądrze sterowników do IDE. Może też to się zdarzyć, jeśli w menadżerze uruchamiania podaliśmy nieprawidłowy parametr root=.

Kernel panic: VFS: Unable to mount root fs on (0:0):

Jeśli wystąpił poprzedni komunikat ten też się pojawi. Jeśli jednak wystąpi sam oznacza to, że system nie mógł sobie poradzić z systemem plików, czyli nie wkompilowaliśmy w jądro obsługi systemu plików, jakich używamy, np. Ext2/3, ewentualnie wkompilowaliśmy go jako moduł i stąd problemy.

Kernel panic: No init found. Try passing init= option to kernel:

Przyczyna tego komunikatu zazwyczaj jest taka sama jak poprzednio. System plików nie został zamontowany prawidłowo i system nie mógł odnaleźć komendy init.

...FATAL: Module sd_mod not found...

Komunikat ten nie jest szczególnie groźny, jeśli nie zapomnieliśmy wybrać niezbędnych opcji z jądra. Oznacza on, iż taki moduł nie został znaleziony i nie może zostać uruchomiony. Przyczyną teko może być to, iż wkompilowaliśmy moduł na stałe a jego wpis pozostał w pliku /etc/modprobe.conf lub też nie wkompilowaliśmy w ogóle. W takim wypadku (o ile nie będziemy wykorzystywać już tego modułu), możemy skasować odnoszący się do niego wpis.

Dziękujemy za uwagę!