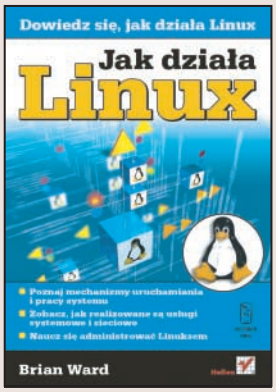
Jak działa Linux



Autor: Brian Ward
Tłumaczenie: Wojciech Moch
ISBN: 83-7361-753-1
Tytuł oryginału: How Linux Works
Format: B5, stron: 392

Jeśli chcesz poznać Linuksa „od środka” – sięgnij po tę książkę

Książka „Jak działa Linux” zawiera wszystkie informacje dotyczące wnętrza systemu Linux oraz zasad korzystania z niego. Przedstawia zarówno zagadnienia podstawowe – przebieg procesu uruchamiania systemu i mechanizmy obsługi sieci, jak i tematy zaawansowane, związane z administracją systemem, jego współpracą z urządzeniami zewnętrznymi oraz programowaniem.

- Podstawy korzystania z Linuksa – polecenia, struktura plików i katalogów oraz dyski logiczne
- Przebieg procesu uruchamiania Linuksa
- Konfigurowanie i uruchamianie sieci i usług sieciowych
- Pisanie własnych programów w języku powłoki oraz w C++
- Kompilowanie programów
- Konserwacja jądra systemu
- Obsługa urządzeń peryferyjnych
- Tworzenie kopii bezpieczeństwa

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

Rozdział 3.
Jak uruchamia się Linux?

Znamy już fizyczną strukturę systemu Linux, wiemy, czym jest jądro, znamy też zasady pracy z procesami. W niniejszym rozdziale zajmiemy się uruchamianiem systemu, czyli mechanizmami ładującymi jądro do pamięci i uruchamiającymi podstawowe procesy systemowe.

Jak się okazuje, sam proces uruchamiania systemu jest bardzo prosty:

1. Program rozruchowy znajduje na dysku obraz jądra, ładuje go do pamięci i uruchamia.
2. Jądro inicjalizuje urządzenia i ich sterowniki.
3. Jądro montuje system plików partycji podstawowej.
4. Jądro uruchamia program o nazwie *init*.
5. Program *init* uruchamia pozostałe procesy.
6. Ostatnie procesy uruchamiane przez program *init* w ramach rozruchu systemu pozwalają na zalogowanie użytkowników.

Rozpoznawanie każdego z etapów procesu rozruchu systemu jest niezwykle istotne przy rozwiązywaniu problemów występujących w czasie rozruchu, jak również pozwala na dokładne poznanie całości systemu. Na początek zajmiemy się programem rozruchowym (*boot loader*), który po włączeniu komputera i zakończeniu wszystkich procedur samosprawdzania sprzętu wyświetla pierwsze komunikaty albo zadaje pytanie o to, który system operacyjny chcemy uruchomić. Gdy dokonamy już wyboru, program rozruchowy ładuje do pamięci jądro systemu Linux i przekazuje do niego sterowanie.

Samo jądro zostało szczegółowo opisywane w innej części książki (podrozdział 2.2 objaśnia rolę jądra w systemie, a w rozdziale 10. dowiemy się, jak można zbudować swoje własne jądro). W niniejszym rozdziale opisany zostanie jedynie etap inicjalizacji jądra, czyli etap, w którym jądro wypisuje mnóstwo komunikatów opisujących sprzęt znajdujący w naszym komputerze. Po wyświetleniu komunikatu mówiącego o tym, że jądro zamontowało system plików partycji podstawowej (*/*), uruchamiany jest program *init*:

```
VFS: Mounted root (ext2 filesystem) readonly.
```

Chwilę później pojawi się komunikat o uruchamianiu programu *init*, a później komunikaty o uruchamianiu usług systemowych, a na końcu wyświetlony zostanie symbol zachęty logowania.

W systemach Red Hat Linux pierwszy komunikat programu *init* jest trudny do przeoczenia, ponieważ „wita" on nas w systemie „Red Hat Linux". Wszystkie późniejsze komunikaty uzupełniane są informacją o sukcesie lub porażce poszczególnych operacji podawaną w nawiasach kwadratowych po prawej stronie ekranu.

Większa część tego rozdziału poświęcona jest programowi *init*, ponieważ jest on częścią procesu rozruchu systemu, nad którą mamy największą kontrolę. Dopiero w podrozdziale 3.2 zajmiemy się programami rozruchowymi.

3.1. Program init

W programie *init* nie ma nic wyjątkowego. Przechowywany jest w katalogu */sbin* razem z innymi programami systemu Linux, od których praktycznie niczym się nie różni. Głównym zadaniem tego programu jest uruchamianie i zatrzymywanie innych programów w zadanej kolejności.

Istnieje kilka różnych wersji programu *init*, a większość systemów linuksowych korzysta z (opisywanego w tej książce) programu tworzonego w stylu systemu System V. Niektóre dystrybucje używają prostszych wersji programu *init* przypominających te stosowane w systemach BSD, ale są one rzadko spotykane.

3.1.1. Poziomy uruchomienia (runlevels)

W każdym momencie działania systemu Linux uruchomiony jest pewien podstawowy zbiór procesów. Taki stan maszyny nazywany jest *poziomem uruchomienia* (*runlevel*) i oznaczany numerem od 0 do 6. Sam system większość czasu spędza na tylko jednym spośród poziomów uruchomienia, jednak w czasie zamykania systemu program *init* przełącza się na inny poziom uruchomienia w celu prawidłowego zakończenia działania usług systemowych i zatrzymania samego jądra systemu. Jeszcze inny poziom uruchomienia przeznaczony jest dla *trybu pojedynczego użytkownika* (ang. *single user*) opisywanego w punkcie 3.2.4.

Najprostszą metodą poznania poszczególnych poziomów uruchomienia jest przeglądanie pliku konfiguracyjnego */etc/inittab*. Spróbujmy wyszukać w nim następujący wiersz:

```
id:5:initdefault:
```

Zapis w tym wierszu oznacza, że domyślnym poziomem uruchomienia będzie poziom 5. Wszystkie wiersze w pliku *inittab* mają tę samą postać. Składają się one z czterech pól oddzielanych dwukropkami ułożonych w następującej kolejności:

- Niepowtarzalny identyfikator (krótki ciąg znaków, taki jak ciąg `id` z powyższego przykładu).
- Stosowane poziomy uruchomienia.
- Akcja podejmowana przez program *init* (w powyższym przykładzie akcją tą było ustalenie numeru domyślnego poziomu uruchomienia na `5`).
- Polecenie do uruchomienia (opcjonalne).

W powyższym przykładzie nie jest podane żadne polecenie do wykonania, ponieważ akcja `initdefault`, czyli ustalenie domyślnego poziomu uruchomienia, nie wiąże się z wywołaniem żadnego konkretnego polecenia. Nieco dalej w pliku *initab* z pewnością znajdziemy linię podobną do poniższej:

```
l5:5:wait:/etc/rc.d/rc 5
```

Jest to bardzo ważny wiersz, ponieważ wywołuje on większość konfiguracji i usług systemowych zapisanych w katalogach *rc*.d* i *init.d* (więcej na ten temat w punkcie 3.1.2). Z tego wiersza wynika, że na piątym poziomie uruchomienia wykonywane jest polecenie o nazwie `/etc/rc.d/rc 5`. Na podstawie akcji `wait` możemy określić, kiedy i jak program *init* uruchomi to polecenie: po przejściu na piąty poziom uruchomienia polecenie `rc 5` ma być uruchomione jednokrotnie, a następnie program *init* ma czekać przed podjęciem kolejnych działań.

Poza akcjami `initdefault` i `wait` dostępnych jest jeszcze kilka innych. Wiele z nich dotyczy zarządzania zasilaniem, a wszystkie opisane zostały na stronie podręcznika *inittab*(5). W kolejnych podpunktach zajmiemy się tymi akcjami, które są wykorzystywane najczęściej.

respawn

Akcja `respawn` powoduje, że program *init* uruchomi podane polecenie, a po jego zakończeniu uruchomi je ponownie. W pliku *inittab* najprawdopodobniej znajdziemy zapis podobny do poniższego:

```
1:2345:respawn:/sbin/mingetty tty1
```

Program *getty* zajmuje się udostępnianiem znaków zachęty logowania. Powyższy wiersz dotyczy pierwszej konsoli wirtualnej (*/dev/tty1*), którą możemy wywołać, naciskając klawisze *Alt+F1* lub *Ctrl+Alt+F1* (więcej na ten temat w podrozdziale 3.3). Akcja `res➥pawn` powoduje, że po wylogowaniu się z systemu symbol zachęty logowania ponownie pojawia się na ekranie.

ctrlaltdel

Akcja `crtlaltdel` definiuje zachowanie systemu po naciśnięciu w konsoli wirtualnej klawiszy *Ctrl+Alt+Del*. W większości systemów wywoływane jest wtedy polecenie powodujące ponowne uruchomienie systemu wykorzystujące polecenie `shutdown`, opisane w punkcie 3.1.5.

sysinit

Akcja `sysinit` określa pierwsze polecenie uruchamiane przez program *init* w czasie uruchamiania systemu jeszcze przed wejściem na jakikolwiek poziom uruchomienia.

3.1.2. Jak uruchamiane są procesy na różnych poziomach uruchomienia?

Teraz możemy zacząć mówić o tym, jak program *init* uruchamia usługi systemowe, jeszcze zanim pozwoli się nam zalogować do systemu. Proszę sobie przypomnieć przedstawiany wcześniej wiersz z pliku *inittab*:

```
l5:5:wait:/etc/rc.d/rc 5
```

Ten niewielki kawałek tekstu powoduje uruchomienie bardzo wielu programów. Nazwa `rc` jest skrótem od *run commands*, czyli *uruchom polecenia*, ale wiele osób mówi o tych poleceniach jak o skryptach, programach albo usługach. Gdzie w takim razie są te polecenia?

W naszym przykładzie, dla piątego poziomu uruchomienia, wszystkie polecenia najprawdopodobniej znajdują się w katalogu */etc/rc.d/rc5.d* lub */etc/rc5.d*. Poziom uruchomienia `1` korzysta z katalogu *rc1.d*, poziom uruchomienia `2` — z katalogu *rc2.d* i tak dalej. W katalogu *rc5.d* najprawdopodobniej znajdziemy następujące pliki:

```
S10sysklogd     S20ppp          S99gpm
S12kerneld      S25netstd_nfs   S99httpd
S15netstd_init  S30netstd_misc  S99rmnologin
S18netbase      S45pcmcia       S99sshd
S20acct         S89atd
S20logoutd      S89cron
```

Polecenie `rc 5` uruchamia programy znajdujące się w katalogu podanego poziomu uruchomienia, wywołując następujące polecenia:

```
S10sysklogd start
S12kerneld start
S15netstd_init start
S18netbase start
...
...
S99sshd start
```

Proszę zwrócić uwagę na parametr `start` podawany do każdego z tych poleceń. Litera `S` znajdująca się na początku nazwy każdego z tych poleceń oznacza, że polecenie powinno być uruchamiane w trybie startu, natomiast liczba (od `00` do `99`) oznacza kolejność uruchamiania polecenia przez program `rc`.

Polecenia *rc*.d* są najczęściej skryptami powłoki uruchamiającymi programy zapisane w katalogach */sbin* lub */usr/sbin*. Najczęściej możliwe jest sprawdzenie operacji wykonywanych przez te polecenia, przeglądając ich treść w programie `less` lub innym.

Możliwe jest też ręczne uruchamianie tych usług. Na przykład, jeżeli chcielibyśmy ręcznie uruchomić serwer WWW *httpd*, to wystarczyłoby wywołać polecenie `S99httpd start`. Podobnie, jeżeli kiedykolwiek zajdzie potrzeba zakończenia pracy jednej z tych usług w czasie normalnej pracy komputera, możliwe jest uruchomienie odpowiedniego polecenia z katalogu *rc*.d* z parametrem `stop` (na przykład `S99httpd stop`).

Niektóre z katalogów *rc*.d* zawierają polecenia, których nazwy zaczynają się od litery `K` (od słowa *kill*, czyli „zabij"). W takim przypadku skrypt *rc* uruchamia te polecenia z parametrem `stop`, a nie `start`. Najbardziej prawdopodobne jest znalezienie takich poleceń w katalogach poziomów uruchomienia związanych z zamykaniem systemu.

3.1.3. Dodawanie i usuwanie usług

Jeżeli chcielibyśmy dodać, usunąć lub zmodyfikować zestaw usług zapisanych w katalogach *rc*.d*, to musimy najpierw dokładniej przyjrzeć się tym katalogom. Wywołanie pełnych danych katalogu powinno wyświetlić mniej więcej takie dane:

```
lrwxrwxrwx  .  .  . S10sysklogd -> ../init.d/sysklogd
lrwxrwxrwx  .  .  . S12kerneld -> ../init.d/kerneld
lrwxrwxrwx  .  .  . S15netstd_init -> ../init.d/netstd_init
lrwxrwxrwx  .  .  . S18netbase -> ../init.d/netbase
...
```

Polecenia znajdujące się w katalogu *rc*.d* są tak naprawdę dowiązaniami symbolicznymi do plików znajdujących się w katalogu *init.d*, który najczęściej znajduje się w katalogu */etc* lub */etc/rc.d*. Dystrybucje Linuksa korzystają z tych dowiązań, dzięki którym możliwe jest korzystanie z tych samych skryptów niezależnie od poziomu uruchomienia. Ta konwencja nie jest żadnym wymogiem, ale bardzo ułatwia organizację pracy.

Pierwszym pomysłem na zablokowanie uruchamiania jednego z poleceń z katalogu *init.d* na danym poziomie uruchomienia może być usunięcie odpowiadającego mu dowiązania symbolicznego z jednego z katalogów *rc*.d*. To z całą pewnością zadziała, ale jeżeli kiedykolwiek zostaniemy zmuszeni ponownie włączyć to polecenie do sekwencji automatycznego uruchamiania, to będziemy mieć kłopoty z przypomnieniem sobie pełnej nazwy dowiązania. Z tego powodu nie należy usuwać dowiązań symbolicznych z katalogów *rc*.d*, ale raczej dodawać znak podkreślenia (_) na początku ich nazw:

```
mv S99httpd _S99httpd
```

W czasie uruchamiania systemu program *rc* zignoruje nazwy w rodzaju *_S99httpd*, ponieważ nie zaczynają się one od litery `S` lub `K`. Co więcej, nadal mamy dostęp do pełnej nazwy skrótu, a także do polecenia przez niego wskazywanego, dzięki czemu możemy je szybko uruchomić ręcznie, jeżeli zajdzie taka potrzeba.

Dodając usługę do sekwencji automatycznego uruchamiania, musimy przygotować nowy skrypt w katalogu *init.d*, a następnie przygotować dowiązanie symboliczne w odpowiednim katalogu *rc*.d*. Osoby nieznające języka skryptów powinny najpierw przeczytać rozdział 7. Najprostszą metodą na napisanie takiego skryptu jest przejrzenie skryptów znajdujących się w katalogu *init.d*, skopiowanie jednego z nich (najlepiej pasującego do naszych wymagań) i zmodyfikowanie takiej kopii.

Należy się też upewnić, że nową usługę umieściliśmy w odpowiednim miejscu w sekwencji uruchamianych usług. Jeżeli zostanie ona uruchomiona zbyt wcześnie, może nie działać właściwie ze względu na zależności wiążące ją z innymi usługami. Mało ważne usługi najczęściej otrzymują od administratorów numery powyżej 90 i są uruchamiane po wszystkich usługach dostarczonych razem z systemem.

Dystrybucje Linuksa wyposażone są najczęściej w specjalne polecenie włączające i wyłączające usługi zapisane w katalogach *rc*.d*. Na przykład w dystrybucji Debian jest to polecenie `update-rc.d`, a w dystrybucji Red Hat — `chkconfig`. Dostępne są również odpowiednie programy z interfejsem graficznym. Korzystanie z tych programów ułatwia zachowanie spójności katalogów startowych, a także pozwala na łatwe dokonywanie uaktualnień systemu.

Jednym z najczęstszych problemów z instalowaniem Linuksa jest nieprawidłowa konfiguracja serwera XFree86 powodująca migotanie obrazu, w wyniku czego systemu nie da się używać. Poprawienie tego zachowania wymaga uruchomienia systemu w trybie pojedynczego użytkownika i zmodyfikowanie naszego poziomu uruchomienia i związanych z nim usług. W katalogach *rc*.d* lub pliku */etc/inittab* należy szukać zapisów zawierających ciągi znaków `xdm`, `gdm` lub `kdm`.

3.1.4. Kontrolowanie programu init

Czasami konieczne jest nakazanie programowi *init* wykonania zmiany poziomu uruchomienia, ponowne odczytanie pliku *inittab* albo zamknięcie systemu. Identyfikator procesu *init* ma zawsze wartość jeden, ponieważ jest to pierwszy program uruchamiany w systemie.

Do kontrolowania działania programu *init* służy polecenie `telinit`. Na przykład przełączenie na trzeci poziom uruchamiania wykonywane jest poniższym poleceniem:

```
telinit 3
```

W czasie przełączania poziomów uruchomienia program *init* próbuje zakończyć wszystkie procesy, które nie występują w pliku *inittab* w opisie nowego poziomu. Należy więc być bardzo ostrożnym przy wykonywaniu tej operacji.

Po wprowadzeniu jakiejkolwiek zmiany do pliku *inittab* musimy poinformować program *init* o wprowadzonych zmianach i zmusić go do ponownego odczytania tego pliku. Niektórzy wywołują w tym celu polecenie `kill -HUP 1`. Jest to tradycyjna metoda, która funkcjonuje w większości wersji Uniksa, o ile polecenie zostanie wpisane prawidłowo. To samo działanie ma też znacznie prostsze polecenie:

```
telinit q
```

Poza tym polecenie `telinit s` umożliwia przejście do trybu pojedynczego użytkownika (więcej na ten temat zawarto w punkcie 3.2.4).

3.1.5. Wyłączanie systemu

Program *init* kontroluje też proces zamykania i ponownego uruchamiania systemu. W Linuksie prawidłową metodą zamknięcia systemu jest wywołanie polecenia `shutdown`.

Możliwe są dwa podstawowe sposoby wykorzystania polecenia `shutdown`. Po pierwsze, jeżeli chcemy zatrzymać system, pozwala ono na zatrzymanie maszyny i wyłączenie komputera. Aby wykonać natychmiastowe zatrzymanie, należy wpisać poniższe polecenie:

```
shutdown -h now
```

W większości nowych komputerów, na których zainstalowane są w miarę nowe wersje Linuksa, takie zatrzymanie systemu powoduje też automatyczne wyłączenie zasilania komputera. Dostępna jest też operacja ponownego uruchomienia systemu. W tym celu zamiast opcji `-h` należy podać opcję `-r`.

Proces wyłączania systemu trwa kilka sekund. W czasie jego trwania nie wolno wyłączać ani ponownie uruchamiać komputera. W poprzednim przykładzie wartość `now` („teraz") oznacza moment, w którym ma nastąpić wyłączenie. Ten parametr musi być podawany zawsze, choć istnieje wiele sposobów określenia czasu wyłączenia. Jeżeli chcemy, żeby komputer wyłączył się dopiero za jakiś czas, to można podać w tym parametrze wartość *+n*, gdzie *n* jest liczbą minut jaką polecenie `shutdown` ma odczekać, zanim zacznie wyłączać system. Pozostałe opcje polecenia opisywane są na stronie podręcznika *shutdown*(8).

Poniższe polecenie powoduje ponowne uruchomienie systemu po dziesięciu minutach od wywołania polecenia:

```
shutdown -r +10
```

W Linuksie polecenie `shutdown` zawiadamia wszystkich zalogowanych użytkowników o planowanym zamykaniu komputera, choć nie robi nic ponad to. Jeżeli podamy czas zamknięcia systemu inny niż `now`, to polecenie `shutdown` utworzy plik */etc/nologin*. Obecność tego pliku uniemożliwia użytkownikom (z wyjątkiem użytkownika *root*) zalogowanie się do systemu.

Gdy w końcu nadejdzie moment wyłączenia komputera, polecenie `shutdown` nakazuje programowi *init* wykonanie przełączenia na poziom uruchomienia `0` (jeżeli system ma być wyłączony) lub `6` (jeżeli wykonywane będzie ponowne uruchomienie). Po wejściu do poziomu uruchomienia `0` lub `6` wykonywane są podane niżej operacje. Można się o tym przekonać, przeglądając skrypty znajdujące się w katalogach *rc0.d* i *rc6.d*.

1. Program *init* kończy działanie wszystkich dostępnych procesów (podobnie działa przy przełączaniu w dowolny inny poziom uruchomienia).
2. Uruchamiane są początkowe polecenia z katalogu *rc0.d* lub *rc6.d* blokujące pliki systemowe i wykonujące inne przygotowania do wyłączenia.
3. Uruchamiane są kolejne polecenia z katalogu *rc0.d* lub *rc6.d* odmontowujące wszystkie systemy plików za wyjątkiem partycji podstawowej.

4. Następne polecenia z katalogu *rc0.d* lub *rc6.d* ponownie montują partycję podstawową w trybie *tylko do odczytu*.
5. Dalsze polecenia z katalogu *rc0.d* lub *rc6.d* zapisują poleceniem `sync` wszystkie buforowane dane do systemu plików (była o tym mowa w punkcie 2.4.4).
6. Ostatnie polecenia z katalogu *rc0.d* lub *rc6.d*, nakazują jądru zatrzymać się lub ponownie się uruchomić, wywołując odpowiednio polecenie `reboot`, `halt` lub `poweroff`.

Programy `reboot` i `halt` zachowują się inaczej w zależności od poziomu uruchomienia, na którym są one wywoływane, co może powodować pewne nieporozumienia. Domyślnie programy te wywołują polecenie `shutdown` z opcją `-r` lub `-h`. Jeżeli jednak system znajduje się w poziomie uruchomienia `0` lub `6` to oba te polecenia nakazują jądru systemu natychmiastowe zatrzymanie. Jeżeli musimy zamknąć system w pośpiechu (nie zważając na ewentualne uszkodzenia wynikające z nieprawidłowego zamknięcia), to w obu programach można zastosować opcję `-f`.

3.2. Programy rozruchowe

Zanim jądro będzie mogło uruchomić program *init*, samo musi być uruchomione przez program rozruchowy (*boot loader*). Czasami zdarza się, że będziemy musieli nakazać programowi rozruchowemu załadowanie innego jądra lub systemu operacyjnego albo uruchomienie się w innym trybie. Jest to szczególnie ważne wtedy, gdy próbujemy naprawić system, w którym występują problemy uniemożliwiające prawidłowe uruchomienie. Rozwiązane takich problemów może być wykonane w trybie pojedynczego użytkownika albo za pomocą innego jądra systemu.

Program rozruchowy ładuje do pamięci obraz jądra i przekazuje mu kontrolę nad procesorem, być może przekazując przy tym kilka parametrów. Parametry te są zwykłymi ciągami znaków, na przykład `-s` oznacza uruchomienie w trybie pojedynczego użytkownika, a `root=`*`partycja`* nakazuje jądru używanie partycji określonej za pomocą parametru *`partycja`* jako partycji podstawowej. W parametrze jądra można podać też poziom uruchomienia, na którym jądro ma zatrzymać proces uruchamiania systemu.

Najpierw jednak musimy wiedzieć, jak dostać się do wiersza poleceń programu rozruchowego. Niestety, dostępnych jest kilka rodzajów programów rozruchowych, a ponieważ pozwalają one na zmiany swojego zachowania, dystrybucje Linuksa w pełni korzystają z tych możliwości.

W następnych punktach dowiemy się, jak można dostać się do wiersza poleceń programu rozruchowego w celu wprowadzenia nazwy uruchamianego jądra i jego parametrów. Każdy, kto chciałby wiedzieć, jak można zainstalować program rozruchowy albo zmienić jego konfigurację, powinien zajrzeć do podrozdziału 10.5.

3.2.1. LILO

Program rozruchowy LILO wykorzystywany jest niemal od czasu pojawienia się pierwszego jądra Linuksa. Wiersz poleceń LILO najczęściej kończy się słowem `boot:`. Jeżeli nasz system uruchamiany jest przez LILO, to najprawdopodobniej przy starcie zobaczymy na ekranie ładny ekran graficzny, ponieważ taką konfigurację wykorzystują niemal wszystkie dystrybucje, włącznie z dystrybucją Red Hat. Na takim ekranie startowym z pewnością znajduje się tekst *Press Control-X for text mode*. Jeżeli zobaczymy taki komunikat, to naciśnięcie klawiszy *Ctrl+X* spowoduje pojawienie się wiersza poleceń programu rozruchowego.

Jeżeli system uruchamiany jest w trybie tekstowym, to wiersza poleceń należy szukać już w momencie, gdy program rozruchowy zgłosi się po raz pierwszy. Jeżeli na ekranie pojawi się tylko napis `LILO`, to wiersz poleceń można wywołać, naciskając klawisz *Shift*. Z drugiej strony, jeżeli wiersz poleceń pojawia się natychmiast po uruchomieniu programu rozruchowego, to musimy działać szybko, gdyż sam system uruchomi się automatycznie, jeżeli przez pewien czas nie będziemy nic wpisywać w wierszu poleceń.

Gdy już znajdziemy się w wierszu poleceń LILO, naciśnięcie klawisza *Tab* spowoduje wypisanie listy opcji jądra i systemu operacyjnego. Domyślna nazwa jądra to najczęściej *linux*, dlatego chcąc uruchomić domyślne jądro bez żadnych opcji, wystarczy, że wpiszemy `linux`. Wszystkie opcje, jakie chcemy przekazać uruchamianemu jądru, należy wpisywać za jego nazwą:

```
linux opcja1 opcja2 ...
```

Na przykład poniższe polecenie uruchamia system w trybie pojedynczego użytkownika:

```
linux -s
```

Jeżeli chcielibyśmy uruchomić system w trybie pojedynczego użytkownika i zastąpić domyślną partycję podstawową partycją */dev/hda3*, należałoby wpisać poniższe polecenie:

```
linux root=/dev/hda3 -s
```

3.2.2. GRUB

Nazwa GRUB jest skrótem od *Grand Unified Bootloader*, czyli *wielki zunifikowany program rozruchowy*. Jest to system rozruchowy powoli wypierający z użycia program rozruchowy LILO. System GRUB może pochwalić się wieloma świetnymi możliwościami, z których najważniejszą jest możliwość przeglądania systemów plików bez konieczności ładowania jądra systemu.

Oczywiście administratorzy systemów Solaris i BSD będą się chwalić, że z takiej możliwości korzystają już od jakiegoś czasu.

GRUB posiada bardzo prosty w obsłudze interfejs menu, jeżeli jednak chcielibyśmy uruchomić system z innym jądrem, zmienić partycję podstawową albo przekazać do jądra dodatkowe opcje, to będziemy musieli skorzystać z minipowłoki. Naciśnięcie w menu klawisza *c* spowoduje pojawienie się wiersza poleceń:

```
grub>
```

Załóżmy, że chcielibyśmy uruchomić system z jądrem znajdującym się w pliku */boot/vmlinuz*, a jako partycję podstawową wykorzystać partycję */dev/hda3*. Co więcej, konfiguracja naszego systemu jest całkowicie zniszczona, więc musimy uruchamiać go w trybie pojedynczego użytkownika (czyli skorzystać opcji `-s`). W tym celu musimy w wierszu poleceń GRUB wpisać następujące polecenia:

```
root (hd0,2)
kernel /boot/vmlinuz root=/dev/hda3 -s
boot
```

Pierwsza linia informuje program rozruchowy o umiejscowieniu partycji podstawowej, czyli systemu plików, w którym GRUB będzie szukał jądra systemu. Zapis `hd0` oznacza pierwszy dysk twardy, czyli pierwszy dysk jaki znajdzie program GRUB. Na przykład w Linuksie oznaczeniem pierwszego dysku jest zapis */dev/hda*. Już jednak dwójka po przecinku oznacza **trzecią** partycję na tym dysku (*/dev/hda3*), ponieważ GRUB numeruje partycje od zera.

Słowo `kernel` pozwala określić umiejscowienie jądra systemu i jego parametry. Ścieżka `/boot/vmlinuz` określa umiejscowienie jądra na partycji `(hd0,2)`. Niestety, standardowo GRUB nie przekazuje do jądra informacji z poprzedniego polecenia `root`, dlatego musimy sami wpisać dodatkowy parametr jądra `root=`*`partycja`*.

Możliwe jest połączenie wierszy `root` i `kernel`. Musimy tylko wpisać przed nazwą obrazu jądra informacje o partycji, na której się ono znajduje. W ten sposób połączone dwa wiersze z powyższego przykładu można zapisać w postaci pojedynczego wiersza: `kernel (hd0,2)/boot/vmlinuz root=/dev/hda3 -s`.

Ostatni wiersz naszego przykładu nakazuje programowi GRUB załadowanie i uruchomienie jądra systemu.

W niektórych systemach (szczególnie wykorzystujących dyski SCSI i wstępnie skonfigurowane jądra) możliwe jest przygotowanie wstępnego RAM-dysku:

```
initrd /boot/initrd
```

W punkcie 10.5.4 będziemy mówić o tym, do czego wykorzystywane są wstępne RAM-dyski.

Jeżeli chcielibyśmy za pomocą programu GRUB uruchomić system z partycji posiadającej własny program rozruchowy (na przykład partycji z systemem Windows), pomocne okażą się poniższe polecenia, w których *`partycja`* oznacza partycję dysku zapisywaną w formacie programu GRUB (na przykład `(hd0,1)`):

```
rootnoverify partycja
makeactive
chainloader +1
boot
```

3.2.3. Inne programy rozruchowe

Istnieje wiele innych metod uruchomienia jądra systemu, w tym również z poziomu systemu DOS, za pomocą programów LOADLIN lub SYSLINUX, poprzez sieć, a nawet bezpośrednio z BIOS-u komputera PC (potrzebny jest do tego tak zwany LinuxBIOS). Większość programów rozruchowych działa na podobnej zasadzie co LILO, chociaż część z nich nie obsługuje klawisza *Tab*, wypisującego opcje jądra. W takich przypadkach odpowiednie informacje można uzyskać z komunikatów diagnostycznych rozruchu jądra. Zazwyczaj jednak opcje jądra wprowadzane są w sposób bardzo podobny do stosowanego w programach LILO i GRUB.

3.2.4. Tryb pojedynczego użytkownika i uruchomienie awaryjne

Jeżeli coś złego przydarzy się w systemie, to pierwszą reakcją administratora jest uruchomienie stabilnego stanu systemu nazywanego **trybem pojedynczego użytkownika** (*single user mode*). W tym trybie system szybko uruchamia powłokę użytkownika *root*, ale w ogóle nie zajmuje się uruchamianiem jakichkolwiek usług. W Linuksie tryb pojedynczego użytkownika jest najczęściej równoznaczny z poziomem uruchomienia `1`. Wejście do trybu pojedynczego użytkownika zazwyczaj chronione jest hasłem użytkownika konta *root*.

Do typowych operacji wykonywanych w trybie pojedynczego użytkownika można zaliczyć:

- Sprawdzanie spójności systemów plików po załamaniu systemu.
- Rozwiązywanie problemów w krytycznych plikach systemowych takich jak */etc/fstab*, */etc/passwd* i */etc/inittab*.
- Odtwarzanie kopii bezpieczeństwa po załamaniu systemu.

W trybie pojedynczego użytkownika nie działa zbyt wiele mechanizmów ułatwiających pracę. Możemy ustalić typ terminala (wprowadzając zmienną `TERM=linux`), uzyskując w ten sposób dostęp do pełnoekranowych edytorów tekstu. Poza tym najprawdopodobniej nie będzie działała sieć. Niestety, konieczne jest ręczne konfigurowanie sieci i innych systemów, a z całą pewnością nie jest to łatwe.

Po zakończeniu pracy w trybie pojedynczego użytkownika możemy wyjść z powłoki i sprawdzić, czy system uruchamia się już normalnie. Jednak zazwyczaj lepszą metodą jest ponowne uruchomienie systemu, ponieważ przejście z trybu pojedynczego użytkownika do normalnego trybu wielodostępu nie zawsze działa należycie.

Jeżeli system został uszkodzony tak bardzo, że system nie uruchamia się nawet w trybie pojedynczego użytkownika, to możemy spróbować jeszcze użyć parametru jądra `-b` i wywołać w ten sposób awaryjne uruchomienie powłoki. Po takim uruchomieniu, partycja podstawowa nie będzie zamontowana w trybie pełnego dostępu, nie będą też uruchomione żadne mechanizmy systemowe. Oznacza to, że zanim będziemy mogli cokolwiek zrobić w systemie, konieczne będzie przemontowanie kilku partycji i zamontowanie

systemu plików */proc*. Jednak w przypadkach aż tak poważnych uszkodzeń najczęściej lepszym rozwiązaniem jest użycie naprawczej płyty CD-ROM. Powłokę awaryjną można też wywołać, podając do jądra parametr `init=/bin/sh`.

W końcu, w przypadku uszkodzenia pliku jądra lub samego programu rozruchowego, bez dodatkowej pomocy nie uda się nam włączyć nawet trybu pojedynczego użytkownika. Najczęściej istnieje możliwość uruchomienia systemu z płyty CD-ROM poprzez wykorzystanie zainstalowanego na niej programu rozruchowego. Wystarczy podać w jego wierszu poleceń opcje wskazujące na naszą partycję podstawową. Tak uruchomiony system może wyglądać nieco dziwnie, ale z pewnością będzie działał i być może umożliwi nam skompilowanie nowego jądra, ratując nas tym samym z opresji.

3.3. Konsole wirtualne

W ostatnim etapie rozruchu systemu uruchamiany jest jeden lub dwa programy pozwalające nam zalogować się do konsoli systemowej. Linux korzysta z dwóch trybów wyświetlania: trybu tekstowego (konsoli) i serwera okien X Window (tryb graficzny najczęściej wykorzystujący menedżera wyświetlania). Jądro uruchamia się normalnie w trybie konsoli, jednak w wielu dystrybucjach system przełącza się w tryb graficzny zaraz po zakończeniu uruchamiania poleceń z katalogów *rc*.d*.

System Linux posiada kilka *konsoli wirtualnych*. Każda z konsoli wirtualnych może działać zarówno w trybie tekstowym, jak i w graficznym. W trybie tekstowym można przełączać się pomiędzy konsolami, stosując kombinacje klawiszy *Alt-klawisz funkcyjny*. Na przykład kombinacja *Alt+F1* włącza konsolę */dev/tty1*, *Alt+F2* — konsolę */dev/tty2* i tak dalej.

Konsole wirtualne korzystające w trybie graficznym z XFree86 działają nieco inaczej. Tutaj nie obowiązują zapisy pobierane z pliku */etc/inittab*, ale system XFree86 przejmuje na własne potrzeby pierwszą wolną konsolę wirtualną. Na przykład, jeżeli proces *getty* działa już na konsolach *tty1* i *tty2*, to nowy serwer XFree86 przejmie na własne potrzeby konsolę *tty3*. Dodatkowo, po uruchomieniu graficznej konsoli przełączanie pomiędzy konsolami następuje po naciśnięciu klawiszy *Ctrl+Alt+klawisz funkcyjny*; dotychczasowa kombinacja *Alt+klawisz funkcyjny* przestaje działać.

W efekcie, jeżeli po uruchomieniu systemu będziemy chcieli skorzystać z konsoli tekstowej, to będziemy musieli nacisnąć klawisze *Ctrl+Alt+F1*. Natomiast, aby powrócić do sesji X11, musimy naciskać po kolei *Alt+F2*, *Alt+F3* i tak dalej, aż w końcu znajdziemy naszą konsolę graficzną.