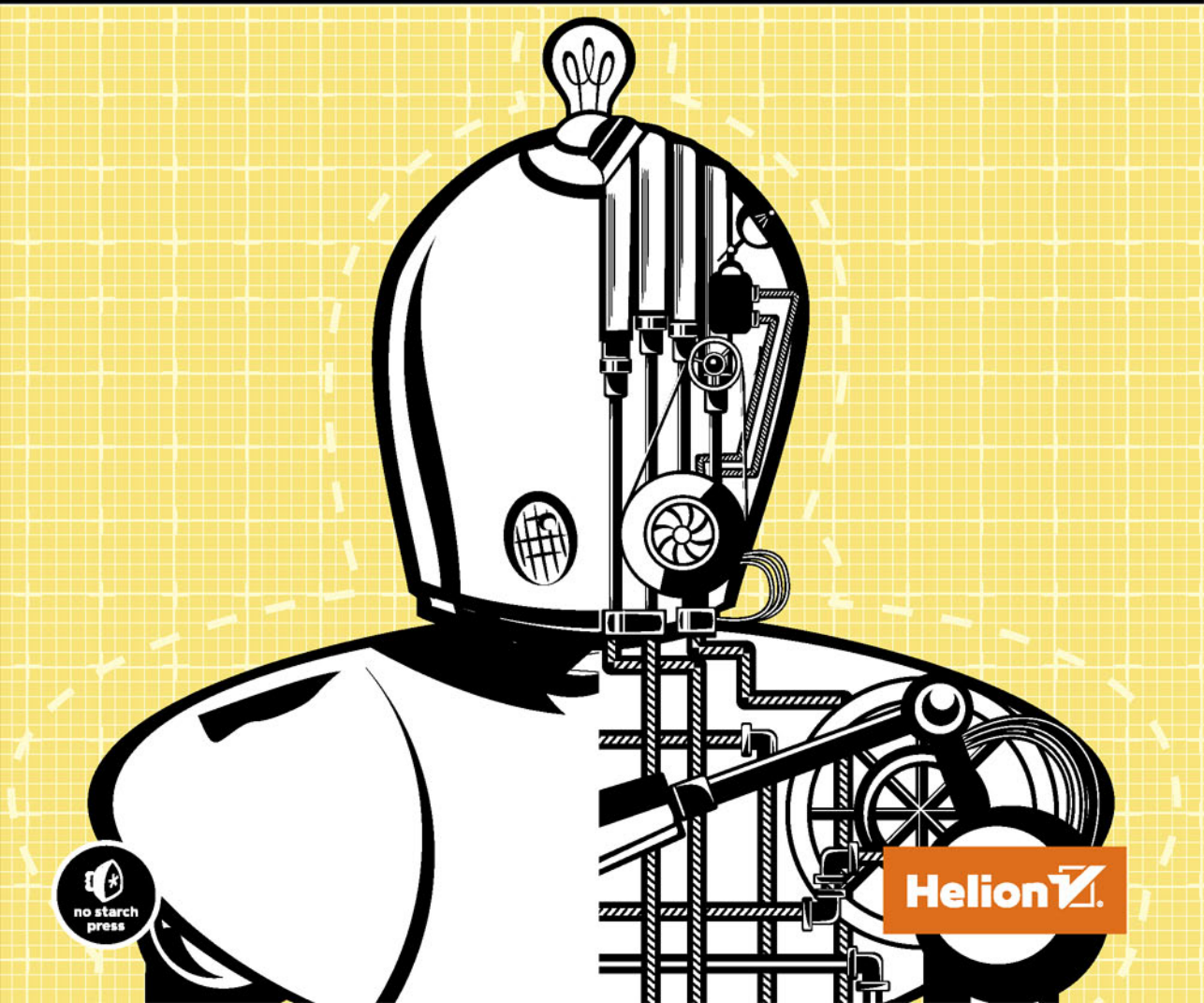
WYDANIE II

JAK DZIAŁA LINUX

PODRĘCZNIK ADMINISTRATORA

BRIAN WARD



no starch press

Helion

Tytuł oryginału: How Linux Works: What Every Superuser Should Know, Second Edition

Tłumaczenie: Piotr Pilch

ISBN: 978-83-283-0980-7

Wydawnictwo HELION
ul. Kościuszki 1c, 44-100 GLIWICE
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *http://helion.pl* (księgarnia internetowa, katalog książek)

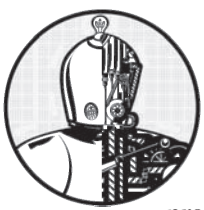
Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
http://helion.pl/user/opinie/jakli2
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Printed in Poland.

Spis treści

5

Jak uruchamia się Linux?



Znasz już fizyczną i logiczną strukturę systemu linuksowego, wiesz, czym jest jądro systemu i jak współpracuje z procesami. W tym rozdziale zaprezentuję procedurę uruchamiania (lub rozruchu) jądra. Innymi słowy, dowiesz się teraz, w jaki sposób jądro przenosi się do pamięci komputera do momentu, w którym uruchomiony zostaje pierwszy proces użytkownika.

Uproszczona procedura uruchamiania systemu wygląda następująco.

1. BIOS komputera lub jego firmware ładuje i uruchamia program rozruchowy (ang. *boot loader*).
2. Program rozruchowy odszukuje na dysku obraz jądra, ładuje go do pamięci i uruchamia.
3. Jądro inicjuje wszystkie urządzenia wraz z ich sterownikami.
4. Jądro montuje podstawowy system plików.
5. Jądro uruchamia program o nazwie `init` związany z procesem o identyfikatorze 1. To w tym momencie *uruchamiana jest przestrzeń użytkownika*.
6. Program `init` wprawia w ruch pozostałe elementy systemu.
7. W pewnym momencie program `init` uruchamia proces pozwalający na zalogowanie się użytkownika. Zazwyczaj dzieje się to pod sam koniec procesu uruchamiania systemu.

W tym rozdziale zajmę się czterema pierwszymi krokami; skoncentruję się na jądrze i programie rozruchowym. W rozdziale 6. podejmę ten temat od momentu uruchomienia przestrzeni użytkownika.

Umiejętność zidentyfikowania poszczególnych etapów uruchamiania systemu okazuje się niezastąpiona podczas rozwiązywania problemów ze startem systemu, ale też ułatwia zrozumienie zasad funkcjonowania systemu jako całości. Niestety domyślne zachowanie w wielu dystrybucjach Linuksa często bardzo utrudnia, a czasem i uniemożliwia zidentyfikowanie pierwszych etapów rozruchu. W efekcie możesz przejrzeć ten proces dopiero po zakończeniu uruchamiania i zalogowaniu się.

5.1. Komunikaty rozruchowe

Tradycyjne systemy uniksowe podczas uruchamiania generują wiele komunikatów diagnostycznych, które dokładnie opisują cały ten proces. Początkowo komunikaty te pochodzą wyłącznie z jądra, ale później pojawiają się też takie, które pochodzą z procesów i procedur inicjujących uruchamianych przez program `init`. Komunikaty te nie są jednak ani ładne, ani spójne, a w niektórych przypadkach nie niosą nawet czytelnych informacji. Większość aktualnych dystrybucji Linuksa stara się jak najlepiej ukrywać je za ekranami startowymi i innymi obrazkami. Dodatkowo rozwijany ciągle sprzęt sprawia, że jądro uruchamia się dzisiaj znacznie szybciej niż dawniej, przez co wszystkie te komunikaty przewijane są tak szybko przez ekran, że nie da się ich przeczytać.

Istnieją dwie metody przeglądania komunikatów diagnostycznych rozruchu jądra i działających później programów. Możemy:

- przejrzeć plik protokołu systemowego jądra; często znajduje się on w pliku */var/log/kern.log*, ale w zależności od konfiguracji systemu może też znaleźć się wśród innych plików protokołów systemowych, w katalogu */var/log/messages* lub innym,
- użyć polecenia `dmesg`, przy czym jego wyjście koniecznie trzeba przepuścić przez program `less`, ponieważ komunikatów na pewno będzie więcej, niż może pomieścić jeden ekran. Polecenie to wykorzystuje bufor cykliczny jądra, który ma ograniczoną wielkość, ale w większości nowoczesnych jąder jest na tyle duży, że przez długi czas może przechowywać komunikaty rozruchowe.

Oto przykład tego, czego można się spodziewać po uruchomieniu polecenia `dmesg`:

```
$ dmesg
[ 0.000000] Initializing cgroup subsys cpu
[ 0.000000] Linux version 3.2.0-67-generic-pae (buildd@toyol) (gcc version 4.6.3 (Ubuntu/Linaro
↪4.6.3-1ubuntu5) ) #101-Ubuntu SMP Tue Jul 15 18:04:54 UTC 2014(Ubuntu 3.2.0-67.101-generic-pae
↪3.2.60)
```

```
[ 0.000000] KERNEL supported cpus:
--ciach--
[ 2.986148] sr0: scsi3-mmc drive: 24x/8x writer dvd-ram cd/rw xa/form2 cdda tray
[ 2.986153] cdrom: Uniform CD-ROM driver Revision: 3.20
[ 2.986316] sr 1:0:0:0: Attached scsi CD-ROM sr0
[ 2.986416] sr 1:0:0:0: Attached scsi generic sg1 type 5
[ 3.007862] sda: sda1 sda2 < sda5 >
[ 3.008658] sd 0:0:0:0: [sda] Attached SCSI disk
--ciach--
```

Po zakończeniu uruchamiania jądra komunikaty generowane są przez procedurę uruchamiania przestrzeni użytkownika. Te komunikaty będą znacznie trudniejsze do przejrzenia i oceny, ponieważ w większości systemów nie są zapisywane do jednego pliku protokołów. Skrypty startowe zazwyczaj wypisują na konsolę swoje komunikaty, a te są zwykle kasowane po zakończeniu procedury uruchamiania. Zazwyczaj nie stanowi to problemu, ponieważ każdy skrypt prowadzi też swój własny protokół. Niektóre wersje programu `init`, takie jak `Upstart` i `systemd`, zarówno w czasie rozruchu, jak i w trakcie normalnej pracy systemu mogą przechwytywać komunikaty, które trafiają jedynie na konsolę.

5.2. Inicjowanie jądra i opcje rozruchu

Podczas rozruchu jądro Linuksa inicjuje się w następującej kolejności.

1. Sprawdzenie procesora.
2. Sprawdzenie pamięci.
3. Rozpoznawanie magistrali urządzeń.
4. Rozpoznawanie urządzeń.
5. Konfigurowanie uzupełniających podsystemów jądra (sieć i tym podobne).
6. Montowanie podstawowego systemu plików.
7. Uruchamianie przestrzeni użytkownika.

Pierwsze kroki nie są szczególnie spektakularne, ale gdy jądro dotrze do rozpoznawania urządzeń, pojawia się pytanie o zależności między nimi. Przykładowo sterowniki urządzenia dyskowego mogą zależeć od funkcji obsługi magistrali oraz obsługi podsystemu SCSI.

Na dalszym etapie procesu inicjacji jądro musi zamontować podstawowy system plików, żeby później uruchomić program `init`. Zwykle nie musimy się przejmować tymi sprawami. Wyjątek może stanowić sytuacja, kiedy niektóre niezbędne komponenty nie są częścią samego jądra, ale dostępne są jako zewnętrzne moduły. Na niektórych komputerach konieczne może być załadowanie tych modułów jeszcze przed zamontowaniem samego systemu plików. Tym problemem oraz jego rozwiązaniem w postaci początkowego systemu plików w pamięci RAM zajmę się w podrozdziale 6.8.

W czasie pisania tej książki jądro nie generowało żadnych komunikatów w związku z przygotowaniami do uruchomienia pierwszego procesu przestrzeni użytkownika. Jednak poniższe komunikaty związane z zarządzaniem pamięcią całkiem dobrze wskazują na to, że zaraz nastąpi przekazanie kontroli do przestrzeni użytkownika, ponieważ w ten sposób jądro zaczyna chronić własną pamięć przed działającymi w niej procesami.

```
Freeing unused kernel memory: 740k freed
Write protecting the kernel text: 5820k
Write protecting the kernel read-only data: 2376k
NX-protecting the kernel data: 4420k
```

W tym miejscu możesz też zobaczyć też komunikat informujący o montowaniu podstawowego systemu plików.

UWAGA *Jeżeli interesujesz się dalszym ciągiem procesu uruchamiania przestrzeni użytkownika realizowanej przez program `init` uruchomiony przez jądro, od razu możesz przejść do rozdziału 6. W pozostałej części tego rozdziału będę opisywał dokładniej kolejne etapy uruchamiania jądra.*

5.3. Parametry jądra

Podczas uruchamiania jądra Linuksa program rozruchowy przekazuje do niego zestaw tekstowych *parametrów jądra*, które decydują o tym, w jaki sposób ma ono zostać uruchomione. Parametry te definiują wiele różnych typów zachowań, takich jak ilość komunikatów diagnostycznych wypisywanych przez jądro, albo podają opcje właściwe dla różnych sterowników.

Parametry jądra użyte przy uruchamianiu swojego systemu można przejrzeć w pliku */proc/cmdline*:

```
$ cat /proc/cmdline
BOOT_IMAGE=/boot/vmlinuz-3.2.0-67-generic-pae root=UUID=70ccd6e7-6ae6-44f6-812c-
↪51aab8036d29 ro quiet splash vt.handoff=7
```

Parametrami mogą być znaczniki składające się z pojedynczego słowa, takie jak `ro` lub `quiet`, ale mogą być nimi również pary *`klucz=wartość`*, takie jak `vt.handoff=7`. Wiele parametrów nie ma większego znaczenia, na przykład parametr `splash` pozwalający wyświetlić ekran startowy, ale jeden z nich jest absolutnie niezbędny — `root`. Definiuje on lokalizację podstawowego systemu plików. Bez niego jądro nie będzie mogło odnaleźć programu `init` i nie uruchomi przestrzeni użytkownika.

Podstawowy system plików możesz zdefiniować za pomocą pliku urządzenia, tak jak w poniższym przykładzie:

```
root=/dev/sda1
```

Jednak w większości nowoczesnych systemów najczęściej używany jest w tym celu identyfikator UUID systemu plików (omawiany w punkcie 4.2.4):

```
root=UUID=70ccd6e7-6ae6-44f6-812c-51aab8036d29
```

Użycie tu parametru `ro` jest całkiem normalne. Nakazuje on, by jądro zamontowało podstawowy system plików w trybie tylko do odczytu w momencie uruchamiania przestrzeni użytkownika. Tryb tylko do odczytu sprawia, że program `fsck` może bezpiecznie skontrolować system plików. Po zakończeniu takiej kontroli proces uruchamiania systemu może ponownie zamontować podstawowy system plików w trybie pełnego dostępu.

Jeżeli jądro Linuksa natknie się na parametr, którego nie rozumie, mimo wszystko go zapamiętuje, a następnie przekazuje jako parametr programu `init` podczas uruchamiania przestrzeni użytkownika. Jeśli na przykład wśród parametrów jądra znajdzie się wartość `-s`, jądro przekaże ją do programu `init`, w wyniku czego ten uruchomi się w trybie pojedynczego użytkownika.

Przyjrzyjmy się teraz mechanice uruchamiania jądra przez program rozruchowy.

5.4. Programy rozruchowe

Na początku procesu rozruchu systemu program rozruchowy (ang. *boot loader*) musi uruchomić jądro. Zadanie programu rozruchowego wydaje się całkiem proste: załadować do pamięci jądro systemu, a następnie uruchomić je, przekazując odpowiednie parametry. Zastanówmy się jednak, z jakimi problemami musi zmierzyć się ten program.

- Gdzie znajduje się jądro?
- Jakie parametry należy przekazać do jądra w momencie jego uruchamiania?

Odpowiedź na te pytania (zazwyczaj) brzmi tak, że jądro oraz niezbędne parametry znajdują się gdzieś w podstawowym systemie plików. Wydaje się, że odszukanie parametrów jądra nie powinno być kłopotliwe, z tym że samo jądro jeszcze nie działa, dlatego nie można przeszukać systemu plików, żeby odnaleźć niezbędne pliki. Co gorsza, niedostępne są też sterowniki urządzeń, z których normalnie korzysta jądro. Wydaje się, że mamy tu do czynienia z problemem typu „jajko czy kura".

Rozważania zacznę od sterowników. W komputerach PC programy rozruchowe uzyskują dostęp do dysków za pośrednictwem interfejsów BIOS (ang. *Basic Input/Output System*) lub UEFI (ang. *Unified Extensible Firmware Interface*). Niemal wszystkie dyski wyposażone są w oprogramowanie pozwalające na dostęp do danych za pomocą adresowania LBA (ang. *Linear Block Addressing*). Mimo że to rozwiązanie ma bardzo niską wydajność, ten tryb adresowania pozwala na uniwersalny dostęp do zawartości dysku. Programy rozruchowe są chyba jedynymi programami, które odwołują się do dysków za pośrednictwem BIOS-u. Samo jądro korzysta już z własnych sterowników o znacznie lepszej wydajności.

Z systemem plików sprawa nie jest już tak prosta. Większość nowoczesnych programów rozruchowych może odczytać tablicę partycji i ma wbudowaną obsługę systemów plików w trybie tylko do odczytu, a to oznacza, że mogą one odczytywać pliki z dysku. Takie możliwości sprawiają, że znacznie łatwiejsze staje się dynamiczne konfigurowanie i rozbudowywanie programu rozruchowego. Starsze linuksowe programy rozruchowe nie miały takich możliwości, przez co ich konfigurowanie sprawiało znacznie większe problemy.

5.4.1. Zadania programu rozruchowego

Oto zadania linuksowego programu rozruchowego.

- Wybieranie spośród kilku różnych jąder.
- Przełączanie pomiędzy różnymi zestawami parametrów jądra.
- Umożliwienie użytkownikowi ręcznej edycji nazw i parametrów jądra systemu (na przykład w celu przejścia do trybu pojedynczego użytkownika).
- Umożliwienie uruchamiania innych systemów operacyjnych.

Od czasu gdy powstało jądro Linuksa, programy rozruchowe stawały się coraz bardziej rozbudowane i przejmowały takie funkcje jak historia i system menu. Jednak największą przewagą ich nowoczesnych wersji jest możliwość elastycznego wyboru jądra i jego parametrów. Co ciekawe, przy okazji okazało się, że pewne wymagania zostały mocno zredukowane. Przykładowo dzięki temu, że możemy teraz wykonać awaryjne uruchomienie systemu częściowo lub całkowicie z napędu podłączonego do portu USB, raczej nie musimy zajmować się ręcznym wprowadzaniem parametrów jądra albo przechodzeniem do trybu pojedynczego użytkownika. Nowoczesne programy rozruchowe dają więcej różnych możliwości niż było to kiedykolwiek możliwe, co bardzo przydaje się osobom tworzącym własne jądra systemu albo próbującym zmieniać ich parametry.

5.4.2. Przegląd programów rozruchowych

Oto najważniejsze programy rozruchowe, z którymi możesz się zetknąć, według ich popularności.

GRUB — praktycznie standardowy program w systemach linuksowych.

LILO — jeden z pierwszych linuksowych programów rozruchowych. ELILO to wersja współpracująca z UEFI.

SYSLINUX — możesz go skonfigurować tak, żeby działał z wieloma różnymi systemami plików.

LOADLIN — uruchamia jądro w systemie MS-DOS.

efilinux — program rozruchowy działający z UEFI, który ma być wzorem i modelem dla innych programów rozruchowych.

coreboot (dawniej **LinuxBIOS**) — zamiennik BIOS-u komputera o wysokiej wydajności, który może zawierać jądro systemu.

Linux Kernel EFISTUB — moduł jądra pozwalający na załadowanie jądra bezpośrednio z partycji systemowej EFI/UEFI. Pojawia się w najnowszych systemach.

W tej książce będziemy się zajmować wyłącznie programem GRUB. Wynika to z faktu, że inne programy rozruchowe są albo łatwiejsze do skonfigurowania niż GRUB, albo są od niego szybsze.

Aby wprowadzić nazwę jądra lub jego parametry, musisz najpierw włączyć wiersz poleceń programu. Niestety czasami nie do końca wiadomo, jak to zrobić, ponieważ dystrybucje Linuksa starają się dopasować do siebie zachowanie i wygląd programu rozruchowego.

W kolejnym podrozdziale opiszę, jak przejść do wiersza poleceń programu rozruchowego w celu wprowadzenia nazwy jądra i jego parametrów. Gdy już będzie wiadomo, jak to zrobić, możesz spróbować samodzielnie skonfigurować i zainstalować swój program rozruchowy.

5.5. Wprowadzenie do programu GRUB

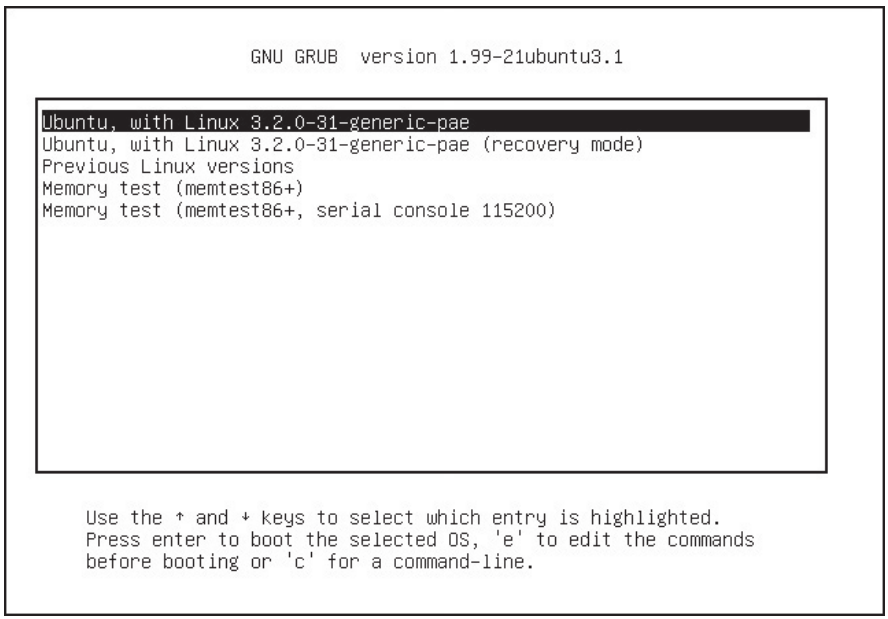
Nazwa *GRUB* jest skrótem od angielskiego *Grand Unified Boot Loader*, czyli „wielki zunifikowany program rozruchowy". Omówię tutaj wersję GRUB 2. Czasami używana jest jeszcze starsza wersja nazywana teraz GRUB Legacy, która jednak pojawia się bardzo rzadko.

Jedną z najważniejszych funkcji tego programu jest możliwość nawigowania w systemie plików, która umożliwia łatwiejsze wybranie obrazu jądra i jego konfigurację. Przejrzenie menu programu jest dobrą metodą na poznanie rządzących nim zasad. W udostępnianym interfejsie łatwo się poruszać nawet wtedy, kiedy widzisz go po raz pierwszy, co jest całkiem prawdopodobne. W końcu twórcy dystrybucji Linuksa starają się jak mogą, żeby ukryć program rozruchowy przed użytkownikami.

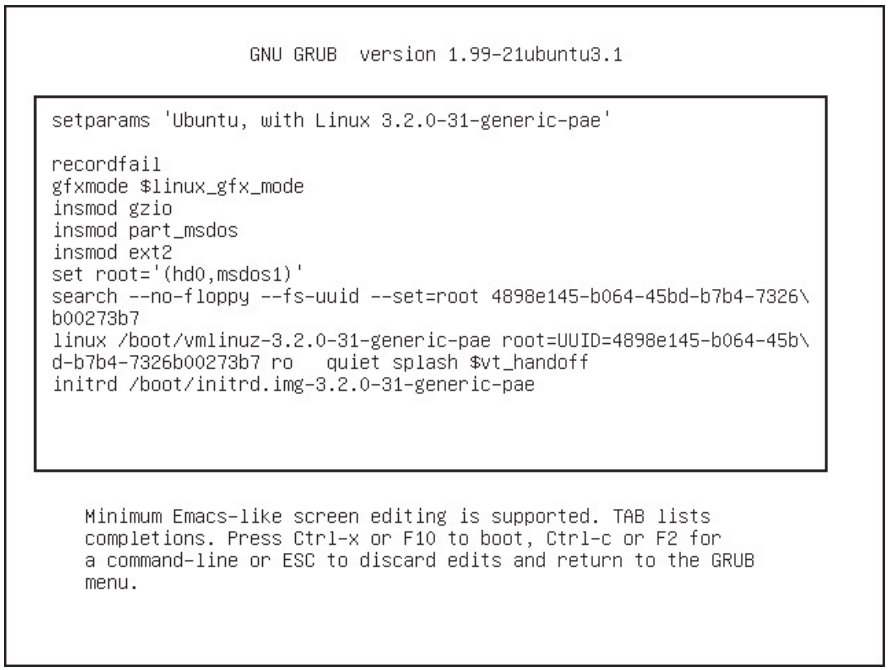
Aby dostać się do menu programu GRUB, należy przytrzymać klawisz *Shift* w czasie, gdy na ekranie widoczne są informacje wyświetlane przez BIOS lub firmware komputera. Jeżeli tego nie zrobisz, program rozruchowy nie zatrzyma się przed załadowaniem jądra do pamięci. Na rysunku 5.1 przedstawiam menu programu GRUB. Po pojawieniu się tego menu można nacisnąć klawisz *Esc*, żeby tymczasowo wyłączyć opóźnienie automatycznego rozruchu.

W ramach poznawania programu rozruchowego proponuję wykonać poniższe kroki.

1. Uruchom (ponownie) swój system linuksowy.
2. Podczas autotestu BIOS-u lub firmware'u komputera przytrzymaj klawisz *Shift*, żeby wywołać menu GRUB-a.
3. Naciśnij klawisz *E*, żeby przejrzeć polecenia konfiguracyjne programu rozruchowego domyślnej opcji rozruchu. Na ekranie powinno pojawić się coś podobnego do przedstawionego na rysunku 5.2.



Rysunek 5.1. Menu programu GRUB



Rysunek 5.2. Edytor konfiguracji programu GRUB

Z tego ekranu dowiesz się, że w tej konfiguracji podstawowy system plików wybierany jest za pomocą identyfikatora UUID, obraz jądra umieszczony jest w pliku */boot/vmlinuz-3.2.0.31-generic-pae*, a wśród parametrów jądra znajdują się opcje `ro`, `quiet`, i `splash`. Jeżeli nie zdarzyło Ci się wcześniej przeglądać tego rodzaju konfiguracji, możesz czuć się trochę nieswojo. Dlaczego odwołanie do podstawowego systemu plików pojawia się tu wielokrotnie i na dodatek z różnymi wartościami? Co oznacza opcja `insmode`? Czy nie jest to funkcja jądra normalnie uruchamiana przy użyciu demona `udevd`?

Takie podwójne zapisy są całkowicie uzasadnione, ponieważ GRUB *nie korzysta* z jądra, a jedynie je *uruchamia*. Widoczna tu konfiguracja składa się wyłącznie z poleceń wewnętrznych samego programu GRUB, a ten jest całkowicie niezależnym światem.

Powstałe zamieszanie wynika z faktu, że GRUB zapożycza terminologię z wielu różnych źródeł. Ma swoje własne „jądro" oraz własne polecenie `insmod` pozwalające na dynamiczne ładowanie modułów programu, które są całkowicie niezależne od jądra Linuksa. Wiele poleceń GRUB-a bardzo przypomina polecenia uniksowej powłoki. Dostępne jest tu nawet polecenie `ls` do wypisywania plików.

Największe zamieszanie powoduje jednak sposób użycia słowa `root`. W ramach wyjaśnienia zaznaczę, że szukając podstawowego systemu plików swojego komputera, należy stosować się do prostej zasady: interesujący nas system plików podawany jest *wyłącznie* w ramach parametrów uruchamianego *jądra*.

W konfiguracji GRUB-a parametry jądra zapisywane są zaraz za nazwą obrazu w poleceniu `linux`. Każde inne wystąpienie słowa `root` dotyczyć będzie jedynie partycji podstawowej GRUB-a. Dla tego programu słowo „root" oznacza system plików, GRUB poszukuje jądra oraz plików obrazów systemów plików umieszczanych w pamięci RAM.

Na rysunku 5.2 partycja podstawowa GRUB-a najpierw ustawiana jest na specjalnym urządzeniu (`hd0,msdos1`). W następnym poleceniu GRUB poszukuje partycji z określonym identyfikatorem UUID. Jeżeli ją znajdzie, właśnie ona stanie się katalogiem podstawowym.

Podsumowując, można stwierdzić, że pierwszy parametr polecenia `linux` (`/boot/vmlinuz-...`) podaje lokalizację pliku z obrazem jądra Linuksa. GRUB załaduje ten plik ze swojej partycji podstawowej. Podobnie działa polecenie `initrd`, z tym że najpierw określa lokalizację wstępnego systemu plików dla pamięci RAM.

Oczywiście możemy dowolnie edytować taką konfigurację w samym programie GRUB. Jest to najprostsza metoda tymczasowego naprawiania uszkodzonego procesu rozruchu systemu. Aby taki problem usunąć trwale, konieczna jest zmiana zapisów konfiguracyjnych (więcej o tym w punkcie 5.5.2), ale na razie przyjrzymy się wewnętrznym elementom GRUB-a; użyjemy przy tym interfejsu wiersza poleceń.

5.5.1. Przeszukiwanie urządzeń i partycji za pomocą wiersza poleceń programu GRUB

Jak można było zobaczyć na rysunku 5.2, GRUB ma własny sposób adresowania urządzeń. Przykładowo pierwszy znaleziony dysk twardy otrzymuje nazwę `hd0`, następny to `hd1` i tak dalej. Niestety przypisanie nazw do urządzeń może podlegać zmianom. Na szczęście GRUB może przeszukać partycje, sprawdzając ich identyfikator UUID, w celu odnalezienia tej, na której znajduje się plik jądra systemu; do tego służy polecenie `search`.

Wypisywanie urządzeń

Aby poznać sposób odwoływania się do urządzeń w systemie przez GRUB, możesz otworzyć jego wiersz poleceń, naciskając klawisz *C* w menu rozruchowym albo w edytorze konfiguracji. Powinien się wtedy pojawić znak zachęty:

```
grub>
```

Możesz tu wprowadzić dowolne z poleceń, które widziałeś w konfiguracji. Na początek jednak proponuję użyć polecenia diagnostycznego `ls`. Bez żadnych parametrów wypisuje ono listę urządzeń znanych GRUB-owi:

```
grub> ls
(hd0) (hd0,msdos1) (hd0,msdos5)
```

W tym przypadku na liście znajduje się jedno urządzenie dyskowe oznaczone jako `(hd0)` oraz dwie partycje — `(hd0,msdos1)` i `(hd0,msdos5)`. Przedrostek `msdos` w nazwie partycji oznacza, że na dysku znajduje się tablica partycji typu MBR. Jeżeli byłaby to tablica partycji typu GPT, nazwy partycji miałyby przedrostek `gpt`. Możliwe są jeszcze kolejne kombinacje z trzecim identyfikatorem, kiedy wewnątrz partycji znajduje się mapa etykiet dysku BSD. Zazwyczaj nie trzeba się tym przejmować, o ile nie pracujesz z kilkoma różnymi systemami operacyjnymi na jednym komputerze.

Bardziej szczegółowe informacje uzyskasz po wprowadzeniu polecenia `ls -l`. Polecenie to może być szczególnie użyteczne, ponieważ podaje też identyfikatory UUID wszystkich partycji na dysku. Oto przykład.

```
grub> ls -l
Device hd0: Not a known filesystem - Total size 426743808 sectors
        Partition hd0,msdos1: Filesystem type ext2 — Last modification time
          2015-09-18 20:45:00 Friday, UUID 4898e145-b064-45bd-b7b4-7326b00273b7 -
Partition start at 2048 - Total size 424644608 sectors
        Partition hd0,msdos5: Not a known filesystem - Partition start at
          424648704 - Total size 2093056 sectors
```

Ten konkretny dysk na pierwszej partycji MBR zawiera linuksowy system plików ext2/3/4, natomiast 5. partycja nosi sygnaturę linuksowej przestrzeni wymiany. Oznacza to dość typową konfigurację dysku. Niestety, patrząc na podany wyżej wydruk, nie da się stwierdzić, że partycja (`hd0,msdos5`) została przeznaczona na przestrzeń wymiany.

Nawigacja wśród plików

Przyjrzyjmy się teraz funkcjom GRUB-a, przy użyciu których możliwe jest poruszanie się w systemie plików. Za pomocą polecenia `echo` możesz sprawdzić podstawową partycję (pamiętaj, że to na niej GRUB spodziewa się znaleźć jądro systemu):

```
grub> echo $root
hd0,msdos1
```

Polecenie `ls` pozwala też na wypisanie listy plików i katalogów w partycji podstawowej. Wystarczy jako parametr wpisać jej nazwę i zakończyć znakiem ukośnika:

```
grub> ls (hd0,msdos1)/
```

Niestety zapamiętanie i późniejsze wpisywanie pełnej nazwy partycji podstawowej może być kłopotliwe, dlatego zastosowanie zmiennej `$root` pozwala na sporą oszczędność czasu i nerwów:

```
grub> ls ($root)/
```

W efekcie zobaczysz krótką listę plików i katalogów umieszczonych w systemie plików tej partycji. Znajdą się na niej takie pozycje jak *etc/*, *bin/* lub *dev/*. Pamiętaj, że to zupełnie osobna funkcja polecenia `ls` z GRUB-a. Wcześniej wypisywało ono urządzenia, tablice partycji i czasami informacje nagłówkowe systemów plików. Teraz pozwala na faktyczne przeglądanie zawartości tych systemów plików.

W podobny sposób możesz też zajrzeć głębiej w hierarchię plików i katalogów danej partycji. Gdy na przykład chcesz zobaczyć zawartość katalogu */boot*, możesz użyć następującego polecenia:

```
grub> ls ($root)/boot
```

UWAGA *Za pomocą klawiszy strzałek w górę i w dół możesz przeglądać zawartość historii poleceń, natomiast klawisze strzałek w lewo i w prawo umożliwiają edycję aktualnie wybranego polecenia. Podobnie działają standardowe kombinacje klawiszy* Ctrl+N *i* Ctrl+P.

Polecenie `set` pozwala na przejrzenie wszystkich zdefiniowanych w danym momencie zmiennych programu GRUB:

```
grub> set
?=0
color_highlight=black/white
color_normal=white/black
--ciach--
prefix=(hd0,msdos1)/boot/grub
root=hd0,msdos1
```

Jedną z najważniejszych zmiennych programu jest zmienna `$prefix` definiująca system plików i katalog, w którym GRUB powinien szukać swojej konfiguracji oraz dodatkowych plików. Więcej informacji na ten temat podam już w następnym punkcie.

Po zapoznaniu się z interfejsem wiersza poleceń GRUB-a możesz wprowadzić polecenie `boot`, żeby uruchomić system z zastosowaniem aktualnej konfiguracji, albo nacisnąć klawisz *Esc*, żeby wrócić do menu programu. Jednak teraz uruchom system, ponieważ zajmiemy się konfigurowaniem GRUB-a, a to najlepiej robić z wykorzystaniem narzędzi oferowanych przez system operacyjny.

5.5.2. Konfigurowanie programu GRUB

Katalog konfiguracji GRUB-a zawiera główny plik konfiguracyjny (*grub.cfg*) oraz wiele modułów ładowanych o rozszerzeniu *.mod*. Wraz z powstawaniem kolejnych wersji programu moduły te były przenoszone do podkatalogów, takich jak *i386-pc*. Zazwyczaj pliki te znajdują się w katalogu */boot/grub* lub */boot/grub2*. Nie należy bezpośrednio modyfikować zawartości pliku *grub.cfg*. Znacznie lepiej będzie, jeżeli użyjesz polecenia `grub-mkconfig` (a w dystrybucji Fedora — `grub2-mkconfig`).

Przeglądanie pliku grub.cfg

Na początek przejrzyjmy zawartość pliku *grub.cfg*, żeby sprawdzić, w jaki sposób GRUB inicjuje swoje menu oraz opcje jądra. Plik ten zawiera różne polecenia GRUB-a, których lista zazwyczaj zaczyna się od pewnych operacji inicjujących. Następnie pojawia się lista pozycji menu dla różnych konfiguracji jądra i procesu jego rozruchu. Sama inicjacja nie jest bardzo skomplikowana. To zaledwie kilka poleceń definicji zmiennych oraz konfiguracji trybu wyświetlania, na przykład takich:

```
if loadfont /usr/share/grub/unicode.pf2 ; then
  set gfxmode=auto
  load_video
  insmod gfxterm
--ciach--
```

Dalej w pliku powinniśmy zobaczyć dostępne konfiguracje rozruchu systemu, przy czym każda z nich będzie zaczynać się od polecenia `menuentry`. Korzystając z tego, czego nauczyłeś się w poprzednim punkcie, z pewnością uda Ci się odczytać i zrozumieć zawartość poniższego przykładu:

```
menuentry 'Ubuntu, with Linux 3.2.0-34-generic-pae' --class ubuntu --class gnu-linux --class gnu
↪--class os {
        recordfail
        gfxmode $linux_gfx_mode
        insmod gzio
        insmod part_msdos
        insmod ext2
        set root='(hd0,msdos1)'
        search --no-floppy --fs-uuid --set=root 70ccd6e7-6ae6-44f6-812c-51aab8036d29
        linux /boot/vmlinuz-3.2.0-34-generic-pae root=UUID=70ccd6e7-6ae6-44f6-812c-51aab8036d29
        ↪ro quiet splash $vt_handoff
        initrd /boot/initrd.img-3.2.0-34-generic-pae
}
```

Przyjrzyj się poleceniom `submenu`. Jeżeli w pliku *grub.cfg* znajduje się wiele poleceń `menuentry`, większość z nich będzie zapewne umieszczona wewnątrz polecenia `submenu`, gromadzącego konfiguracje starszych wersji jądra, tak żeby nie zaśmiecały głównego menu programu.

Generowanie nowego pliku konfiguracyjnego

Jeżeli chcesz wprowadzić zmiany do konfiguracji swojego programu GRUB, raczej nie polecam ręcznego edytowania pliku *grub.cfg*, ponieważ jest on generowany automatycznie i od czasu do czasu system nadpisuje jego zawartość. Nową konfigurację lepiej wprowadzić w innym miejscu, a następnie uruchomić polecenie `grub-mkconfig`, żeby wygenerować nowy plik konfiguracyjny.

Poznawanie metod generowania pliku konfiguracyjnego najlepiej rozpocząć od przyjrzenia się początkowi pliku *grub.cfg*. Powinien tam znajdować się komentarz podobny do poniższego:

```
### BEGIN /etc/grub.d/00_header ###
```

Po dokładniejszym sprawdzeniu tego tropu przekonasz się, że każdy plik w katalogu */etc/grub.d* jest skryptem powłoki, który generuje część zawartości pliku *grub.cfg*. Samo polecenie `grub-mkconfig` jest kolejnym skryptem powłoki, który uruchamia poszczególne pliki z katalogu */etc/grub.d*.

Proponuję wypróbować działanie tego polecenia (nie zapomnij o uprawnieniach superużytkownika). Nie trzeba się przy tym obawiać o zniszczenie aktualnej konfiguracji. Poniższe polecenie wypisuje jedynie wygenerowaną konfigurację na standardowe wyjście:

```
# grub-mkconfig
```

A co zrobić, jeżeli chcesz dodać nowe pozycje do menu albo kolejne polecenia do konfiguracji programu GRUB? Krótko mówiąc, wszystkie takie zmiany umieść w nowym pliku *custom.cfg* znajdującym się w katalogu konfiguracyjnym programu, takim jak */boot/grub/custom.cfg*.

Dłuższa odpowiedź będzie już trochę bardziej złożona. Katalog konfiguracji */etc/grub.d* daje do wyboru dwie opcje: *40_custom* oraz *41_custom*. Pierwsza z nich jest skryptem, który możesz samodzielnie edytować, ale to rozwiązanie mniej stabilne. Każda aktualizacja pakietu może zniszczyć wszystkie zmiany, jakie tu wprowadzisz. Skrypt *41_custom* jest znacznie prostszy i zawiera jedynie serię poleceń ładujących plik *custom.cfg* w momencie uruchamiania GRUB-a. Pamiętaj, że jeżeli zdecydujesz się na tę drugą opcję, Twoje modyfikacje nie pojawią się podczas generowania pliku konfiguracyjnego.

Obie opcje własnych plików konfiguracyjnych nie dają szczególnie wielkich możliwości. W poszczególnych dystrybucjach w katalogu */etc/grub.d* można zobaczyć różne dodatkowe elementy. Przykładowo Ubuntu dodaje do konfiguracji opcję uruchomienia testu pamięci (*memtest86+*).

Aby wygenerować i zapisać nowy plik konfiguracji GRUB-a, należy zastosować opcję `-o` podczas uruchamiania polecenia `grub-mkconfig`, podając lokalizację wynikowego pliku, na przykład tak:

```
# grub-mkconfig -o /boot/grub/grub.cfg
```

Użytkownicy Ubuntu mogą po prostu skorzystać z polecenia `install-grub`. W każdym przypadku uprzednio pamiętaj o wykonaniu kopii bezpieczeństwa dotychczasowego pliku konfiguracyjnego, upewnij się, że instalujesz nowy plik we właściwym katalogu i tym podobne.

Teraz możesz przejść do bardziej technicznych szczegółów GRUB-a i ogólnie programów rozruchowych. Jeżeli czujesz już znużenie tym tematem, możesz od razu przejść do rozdziału 6.

5.5.3. Instalowanie programu GRUB

Instalowanie GRUB-a jest bardziej złożone od jego konfigurowania. Na szczęście zazwyczaj nie musisz przejmować się tym procesem, ponieważ zadanie to przejmuje dystrybucja. Jeżeli jednak próbujesz wykonać kopię dysku z systemem lub odtworzyć taki dysk albo przygotować własną sekwencję rozruchową, całkiem możliwe, że nie obejdzie się bez samodzielnej instalacji programu rozruchowego.

Zanim przejdziemy dalej, proponuję przeczytać punkt 5.8.3, żeby zorientować się w sposobie rozruchu komputera PC i sprawdzić, czy masz do czynienia z procedurą MBR, czy raczej z EFI. Następnie musisz skompilować GRUB-a i określić, w jakim katalogu ma zostać umieszczony. Domyślnie stosuje się katalog */boot/grub*. Jeżeli dystrybucja sama przejmuje to zadanie, możliwe, że samodzielne kompilowanie programu nie będzie konieczne, ale jeżeli będzie trzeba, możesz zajrzeć do rozdziału 16., żeby poznać metody kompilowania oprogramowania z kodu źródło-

wego. Warto się przy tym upewnić, że podczas kompilacji używany jest właściwy cel, ponieważ dla procedur MBR i UEFI stosowane są różne cele kompilacji. Dodatkową różnicą jest wersja procesu EFI — może być 32-bitowa lub 64-bitowa.

Instalowanie GRUB-a w swoim systemie

Instalowanie programu rozruchowego wymaga zdefiniowania elementów, takich jak:

- docelowy katalog programu widzianego przez aktualnie pracujący system; zazwyczaj jest to katalog */boot/grub*, ale jeżeli instalujesz program rozruchowy na innym dysku albo używasz innego systemu, katalog ten może być inny,
- aktualne urządzenie docelowego dysku dla GRUB-a,
- aktualny punkt montowania partycji rozruchowej UEFI, w przypadku procedury rozruchowej UEFI.

Pamiętaj, że GRUB jest systemem modularnym, ale sam program musi odczytać system plików z tym programem, żeby mógł załadować odpowiednie w danej chwili moduły. Twoim zadaniem jest tu skonstruowanie takiej wersji programu, która będzie w stanie odczytać ten system plików, co pozwoli jej na załadowanie pozostałej części swojej konfiguracji (plik *grub.cfg*) oraz wszystkich wymaganych modułów. W Linuksie oznacza to zwykle konieczność zbudowania wersji programu z załadowanym już modułem *ext2.mod*. Po przygotowaniu takiej wersji pozostaje już tylko umieszczenie jej w części dysku przeznaczonej do rozruchu systemu i skopiowanie pozostałych plików do katalogu */boot/grub*.

Na szczęście w zestawie z programem GRUB dostarczane jest narzędzie o nazwie `grub-install` (nie należy go mylić z poleceniem `install-grub` z Ubuntu), które wykonuje większość prac związanych z instalacją plików i konfigurowaniem GRUB-a. Jeśli na przykład aktualny dysk to */dev/sda* i na nim chcesz zainstalować GRUB-a z zastosowaniem standardowego katalogu */boot/grub*, poniższe polecenie zainstaluje program rozruchowy w MBR:

```
# grub-install /dev/sda
```

OSTRZEŻENIE *Nieprawidłowa instalacja GRUB-a może spowodować uszkodzenie sekwencji rozruchu systemu operacyjnego, dlatego nie wykonuj tej operacji pochopnie. Jeżeli masz wątpliwości, lepiej najpierw dowiedz się, jak można wykonać kopię zapasową MBR za pomocą polecenia* `dd`*, wykonaj kopię katalogu aktualnie zainstalowanej wersji GRUB-a i upewnij się, że masz plan na wypadek awarii.*

Instalowanie GRUB-a na zewnętrznym nośniku

Aby zainstalować GRUB na nośniku danych niezależnym od aktualnego systemu, należy zdefiniować katalog GRUB-a na tym urządzeniu, tak jak widzi go aktualnie pracujący system. Załóżmy, że docelowe urządzenie to */dev/sdc*, a katalog podstawowy (lub rozruchowy) systemu plików z tego urządzenia (na przykład */dev/sdc1*)

zamontowany jest w aktualnym systemie w katalog */mnt*. Oznacza to, że podczas instalowania GRUB-a aktualny system będzie widział pliki programu rozruchowego w katalogu */mnt/boot/grub*. A zatem uruchamiając polecenie `grub-install`, musisz zastosować poniższe parametry:

```
# grub-install --boot-directory=/mnt/boot /dev/sdc
```

Instalowanie GRUB-a w systemach z UEFI

Procedura instalacji w systemach z UEFI powinna być prostsza, ponieważ w takiej sytuacji trzeba tylko skopiować program rozruchowy we właściwe miejsce. Oprócz tego konieczne jest jednak „zaprezentowanie" programu rozruchowego firmware'owi komputera za pomocą polecenia `efibootmgr`. Jeżeli polecenie to jest dostępne, program `grub-install` uruchomi je automatycznie, dlatego w teorii instalacja GRUB-a na partycji UEFI wymagać będzie jedynie wydania poniższego polecenia:

```
# grub-install --efi-directory=katalog_efi —-bootloader-id=nazwa
```

W poleceniu tym parametr *`katalog_efi`* opisuje umiejscowienie katalogu UEFI w aktualnym systemie (zazwyczaj jest to katalog */boot/efi/efi*, ponieważ partycja UEFI jest najczęściej montowana w katalogu */boot/efi*), natomiast parametr *`nazwa`* jest prostym identyfikatorem programu rozruchowego, który omówię w punkcie 5.8.2.

Niestety podczas instalowania programu rozruchowego dla UEFI może pojawić się wiele różnych problemów. Przykładowo podczas instalowania go na dysku, który ma się znaleźć w innym systemie, musimy znaleźć sposób na zaprezentowanie nowego programu rozruchowego firmware'owi nowego komputera. Poza tym procedura instalacji na zewnętrznych nośnikach danych wygląda zupełnie inaczej.

Jednak najpoważniejszym problemem jest procedura bezpiecznego rozruchu UEFI (ang. *secure boot*).

5.6. Problemy z bezpiecznym rozruchem UEFI

Jednym z najnowszych problemów, z którymi muszą się zmagać systemy linuksowe, jest funkcja bezpiecznego rozruchu pojawiająca się w najnowszych komputerach PC. Gdy jest aktywna, UEFI wymaga od programu rozruchowego cyfrowego podpisu wystawionego przez zaufaną organizację. Microsoft wymógł na producentach sprzętu dostarczających system Windows 8, żeby zawsze używali funkcji bezpiecznego rozruchu. W efekcie każda próba uruchomienia zainstalowanego, niepodpisanego programu rozruchowego (a jest to powszechne w linuksowych dystrybucjach) skazana jest na niepowodzenie.

Najprostszą metodą obejścia tego problemu dla osób niezainteresowanych używaniem systemów Windows jest wyłączenie funkcji bezpiecznego rozruchu w ustawieniach UEFI. Niestety nie sprawdzi się to w komputerach z dwoma systemami, a poza tym nie jest to rozwiązanie odpowiednie dla wszystkich użytkowników. W związku z tym dystrybucje Linuksa coraz częściej dostarczają podpisane programy rozruchowe. Niektóre rozwiązania są tylko programami wstępnymi dla GRUB-a, ale niektóre to w pełni podpisane sekwencje rozruchu systemu (od programu rozruchowego aż po jądro). Inne rozwiązanie polega na zastosowaniu zupełnie nowego rodzaju programu rozruchowego (programy te w większości bazują na programie *efilinux*).

5.7. Ładowanie innych systemów operacyjnych

UEFI bardzo ułatwia obsługę innych systemów operacyjnych, ponieważ na partycji EFI można zainstalować kilka różnych programów rozruchowych. Niestety takie rozwiązania nie są obsługiwane przez stary dobry rekord MBR, dlatego nawet w komputerach z UEFI możemy być zmuszeni do używania programu rozruchowego współpracującego z MBR. Możliwe jest natomiast takie skonfigurowanie GRUB-a, żeby załadował i uruchomił inny program rozruchowy umieszczony na wyznaczonej partycji. Taki proces nazywany jest *rozruchem łańcuchowym* (ang. *chainloading*).

Aby zastosować rozruch łańcuchowy, należy utworzyć nową pozycję menu w konfiguracji GRUB-a (stosując metody opisane w podpunkcie „Generowanie nowego pliku konfiguracyjnego"). Oto przykładowa konfiguracja uruchamiająca system Windows zainstalowany na trzeciej partycji dysku:

```
menuentry "Windows" {
        insmod chain
        insmod ntfs
        set root=(hd0,3)
        chainloader +1
}
```

Wartość `+1` w opcji `chainloader` nakazuje załadować dowolne dane, jakie znajdują się w pierwszym sektorze tej partycji. Możliwe jest też bezpośrednie załadowanie całego pliku. Przykładowo poniższy wariant umożliwia załadowanie pliku ładującego system MS-DOS, czyli *io.sys*:

```
menuentry "DOS" {
        insmod chain
        insmod fat
        set root=(hd0,3)
        chainloader /io.sys
}
```

5.8. Szczegóły programu rozruchowego

Teraz przyjrzymy się pokrótce wewnętrznym mechanizmom programu rozruchowego. Jeżeli nie interesuje Cię ta tematyka, możesz spokojnie przejść do następnego rozdziału.

Jeśli chcesz zrozumieć zasadę działania takich programów jak GRUB, musisz przyjrzeć się sposobom rozruchu komputera PC po włączeniu zasilania. Z powodu różnych braków tradycyjnych mechanizmów rozruchu komputerów PC istnieje kilka wariantów tego procesu, ale łączą się one w dwa główne schematy — MBR i UEFI.

5.8.1. Rozruch MBR

Oprócz informacji o partycjach opisywanych już w podrozdziale 4.1, rekord rozruchowy MBR (ang. *Master Boot Record*) zawiera niewielki obszar, który jest ładowany do pamięci przez BIOS i uruchamiany zaraz po zakończeniu wstępnego testu (POST — ang. *Power-On Self-Test*). Niestety jest to zbyt mała przestrzeń, żeby zmieścić w niej jakikolwiek program rozruchowy, dlatego wykorzystywane jest dodatkowe miejsce, co tworzy tak zwany *wieloetapowy program rozruchowy*. W tym przypadku pierwszy kawałek kodu z rekordu MBR zajmuje się jedynie załadowaniem pozostałej części kodu programu rozruchowego, który najczęściej zostaje upchnięty pomiędzy rekordem MBR a pierwszą partycją na dysku.

Oczywiście takie rozwiązanie nie jest zbyt bezpieczne, ponieważ każdy program może nadpisać znajdujący się w tym miejscu kod, ale z metody tej korzysta większość programów rozruchowych, w tym i sam GRUB. Co więcej, ta metoda rozruchu nie współpracuje z partycjami GPT, ponieważ tablica GPT umieszczana jest zaraz za rekordem MBR. Mechanizm GPT nie narusza jednak samego MBR w celu zachowania wstecznej zgodności.

Obejściem tego problemu w przypadku partycji GPT jest przygotowanie niewielkiej partycji nazywanej partycją rozruchową o specjalnym identyfikatorze UUID, która daje programowi rozruchowemu dość miejsca na dysku. Partycje GPT są jednak zazwyczaj używane w połączeniu ze schematem UEFI, a nie z tradycyjnym BIOS-em.

5.8.2. Rozruch UEFI

Producenci komputerów PC oraz firmy tworzące oprogramowanie zauważyli, że tradycyjny proces rozruchu związany z BIOS-em jest bardzo ograniczony, dlatego zdecydowano się przygotować nowsze rozwiązanie o angielskiej nazwie *Extensible Firmware Interface* (*EFI*). Trochę czasu upłynęło, zanim EFI przyjęło się na komputerach PC, ale dzisiaj jest już całkiem powszechne. Aktualnym standardem jest UEFI (*Unified* EFI), które udostępnia takie funkcje jak wbudowana powłoka oraz możliwość odczytywania tablic partycji i nawigowania w systemach plików. Sposób partycjonowania GPT jest częścią standardu UEFI.

W systemach z UEFI rozruch wygląda zupełnie inaczej i jest zdecydowanie łatwiejszy do zrozumienia. W tym przypadku wykonywalny kod rozruchowy nie

musi znajdować się poza systemem plików. Na dysku tworzony jest specjalny system plików nazywany *systemową partycją EFI* (ang. *EFI System Partition* — ESP), który zawiera katalog o nazwie *efi*. Każdy program rozruchowy ma swój własny identyfikator oraz podkatalog, taki jak *efi/microsoft*, *efi/apple* albo *efi/grub*. Plik programu rozruchowego musi mieć rozszerzenie *.efi* i znajdować się w jednym z tych katalogów, razem ze wszystkimi plikami wspomagającymi.

UWAGA *Partycja ESP różni się od partycji rozruchowej BIOS opisywanej w punkcie 5.8.1 i ma inny identyfikator UUID.*

I tutaj pojawia się mały problem. Nie można po prostu umieścić kodu starego programu rozruchowego w takim katalogu, ponieważ kod ten został przystosowany do współpracy z interfejsem BIOS-u. Konieczne staje się stosowanie programów rozruchowych przystosowanych do pracy z UEFI. Kiedy przykładowo stosujesz GRUB, musisz użyć specjalnej wersji dla UEFI, a nie dla BIOS-u. Dodatkowo konieczne jest „przedstawienie" takiego programu rozruchowego firmware'owi komputera.

Jak już wspominałem w podrozdziale 5.6, pojawia się też problem „bezpiecznego rozruchu".

5.8.3. Jak działa GRUB?

Podsumuję ten rozdział opisaniem sposobów działania programu GRUB.

1. BIOS lub firmware komputera PC inicjuje elementy sprzętowe i poszukuje kodu rozruchowego na dyskach w kolejności ustalonej w konfiguracji.
2. Po odnalezieniu kodu rozruchowego BIOS lub firmware ładuje go do pamięci i uruchamia. To w tym momencie GRUB ożywa.
3. Ładowane są główne funkcje GRUB-a.
4. Następuje inicjacja. W tym momencie GRUB może już uzyskać dostęp do dysków i systemów plików.
5. GRUB odszukuje swoją partycję podstawową i pobiera z niej konfigurację.
6. GRUB daje użytkownikowi szansę na zmianę tej konfiguracji.
7. Po upłynięciu czasu oczekiwania GRUB wykonuje instrukcje z konfiguracji (sekwencję poleceń opisywaną w punkcie 5.5.2).
8. W trakcie wykonywania instrukcji konfiguracyjnych GRUB może załadować dodatkowy kod (*moduły*) z partycji podstawowej.
9. GRUB wykonuje polecenie `boot`, które ładuje i uruchamia jądro zgodnie z instrukcjami podawanymi w poleceniu `linux`.

Kroki 3. i 4. podanej wyżej sekwencji, w których ładowany jest podstawowy program GRUB-a, mogą być bardzo skomplikowane, co wynika z różnorodności tradycyjnego mechanizmu rozruchu komputerów PC. Podstawowe pytanie brzmi w tym momencie: „Gdzie *jest* jądro GRUB-a?". Można na nie odpowiedzieć na trzy różne sposoby.

- Częściowo może być upchnięte pomiędzy rekordem MBR a początkiem pierwszej partycji.
- Znajduje się na zwyczajnej partycji na dysku.
- Umieszczony jest na specjalnej partycji rozruchowej: partycji rozruchowej GPT, partycji systemowej EFI albo gdzieś indziej.

We wszystkich przypadkach, z wyjątkiem użycia partycji ESP, BIOS komputera załaduje 512 bajtów rekordu MBR i to właśnie od tego zaczyna się praca GRUB-a. Ten niewielki wycinek kodu (pobrany z pliku *boot.img* z katalogu GRUB-a) nie jest jeszcze jego jądrem, ale przechowuje dane o jego lokalizacji i zajmuje się jego załadowaniem do pamięci.

Jeżeli jednak korzystasz z partycji ESP, cały kod GRUB-a umieszczany jest w jednym pliku. Firmware komputera może wtedy przeszukać partycję ESP i bezpośrednio uruchomić jądro GRUB-a albo dowolny inny program rozruchowy, jaki się na niej znajdzie.

Niestety w przypadku większości systemów nie jest to jeszcze pełny obraz sytuacji. Programy rozruchowe mogą wymagać załadowania obrazu wstępnego systemu plików w pamięci RAM i dopiero potem przystąpić do ładowania i uruchamiania jądra systemu operacyjnego. To właśnie takie zachowanie definiuje parametr konfiguracji `initrd` opisywany w podrozdziale 6.8. Zanim jednak zaczniesz poznawać wstępny system plików w pamięci RAM, powinieneś dowiedzieć się, w jaki sposób uruchamiana jest przestrzeń użytkownika. I tym zajmiemy się w następnym rozdziale.

Skorowidz

A

B

C

D

E

F

G

H

L

Ł

M

N

O

P

R

S

Ś

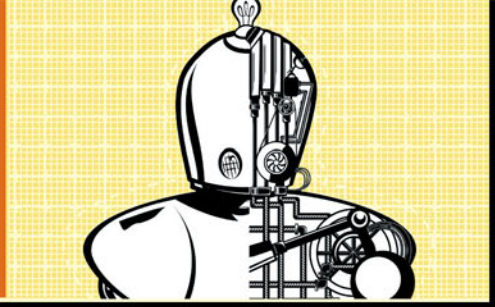
T

U

W

X

Z



WYKORZYSTAJ W PEŁNI SWÓJ KOMPUTER DZIĘKI SYSTEMOWI LINUX!

Linux to system operacyjny będący solą w oku giganta z Redmond. Możliwości, wydajność, elastyczność i bezpieczeństwo Linuksa zostały docenione przez zaawansowanych użytkowników. Natomiast różnorodność powłok graficznych oraz łatwość ich użytkowania przypadły do gustu osobom rozpoczynającym przygodę z tym systemem. Jeżeli chcesz mieć pełną kontrolę nad swoim komputerem i wykorzystać potencjał Linuksa, potrzebujesz tego podręcznika.

Dzięki niemu poznasz wszystkie elementy systemu — począwszy od czeluści jądra, a skończywszy na powłokach użytkownika. W kolejnych rozdziałach znajdziesz informacje na temat podstawowych poleceń, pozwalających na wydajną nawigację w systemie plików oraz wyświetlanie zawartości tych plików, a także na temat narzędzi do kompresji oraz sposobów zarządzania procesami. Ponadto zobaczysz, jak uzyskać dostęp do urządzeń i zarządzać twardymi dyskami. Na tym etapie będziesz już gotów porównać systemy plików ext3 i ext4 oraz poznać kierunki rozwoju systemów plików. Zdobędziesz także ciekawe wiadomości dotyczące startu systemu, konfiguracji harmonogramu zadań, dostępu do sieci oraz tworzenia skryptów ułatwiających pracę. Ta książka będzie pasjonującą lekturą dla każdego użytkownika chcącego zgłębić tajniki pracy z systemem Linux.

Dzięki tej książce:

- poznasz podstawowe polecenia dostępne w systemie Linux
- nauczysz się zarządzać dostępnymi zasobami i procesami
- opanujesz narzędzia programistyczne
- zaznajomisz się z podstawami tworzenia skryptów powłoki
- poznasz system Linux od podszewki

Brian Ward — przygodę z systemem Linux rozpoczął w 1993 roku na komputerze z procesorem 386. Posiada tytuł doktora informatyki, uzyskany na Uniwersytecie Chicagowskim. Jest autorem książek poświęconych Linuksowi. Aktualnie mieszka w San Francisco i pracuje jako konsultant oraz trener.

Helion

37365 numer katalogowy

księgarnia internetowa
http://helion.pl

zamówienia telefoniczne
0 801 339900
0 601 339900

Sprawdź najnowsze promocje:
http://helion.pl/promocje
Książki najchętniej czytane:
http://helion.pl/bestsellery
Zamów informacje o nowościach:
http://helion.pl/nowosci

Helion SA
ul. Kościuszki 1c, 44-100 Gliwice
tel.: 32 230 98 63
e-mail: helion@helion.pl
http://helion.pl

Informatyka w najlepszym wydaniu

sięgnij po WIĘCEJ
KOD KORZYŚCI

ISBN 978-83-283-0980-7

cena: 69,00 zł

no starch press