

Wstęp do systemu operacyjnego UNIX

Laboratorium 4:

Instalacja oprogramowania w systemie Unix

Żaden system operacyjny, tuż po zainstalowaniu nie będzie posiadał funkcjonalności wymaganej przez użytkownika. Zawsze, wcześniej czy później pojawi się konieczność zainstalowania nowego oprogramowania, które może zostać zainstalowane w swoim własnym katalogu, lub doinstalowane do już istniejących katalogów systemowych.

W przypadku konieczności stworzenia w systemie nowego, nie obsługiwanego wcześniej rodzaju systemu plików lub zainstalowania nowego urządzenia konieczne stanie się „dołożenie” nowych fragmentów do jądra. Będą to sterowniki odpowiedzialne za obsługę konkretnych urządzeń. W prostszym przypadku interwencja polegać będzie na dodaniu nowych modułów do jądra. W nieco trudniejszym będziemy musieli dokonać rekompilacji jądra.

0. Podłącz się do systemu jako użytkownik root. Sprawdź czy w instalacji systemu występuje polecenie `snice`.

Ponieważ w naszym systemie przypuszczalnie nie występuje potrzebne nam polecenie skompilujemy i zainstalujemy stosowne oprogramowanie.

1. Przejdź do katalogu `/tmp`. Skopiuj komendą `wget` z adresu URL `http://messy.icsr.agh.edu.pl/sysopy/` do katalogu bieżącego plik `procps-3.1.13.tar.gz`.

2. Rozpakuj plik `procps-3.1.13.tar.gz`. Użyj polecenia `tar` z opjami `xzvf`:

```
1 [bory@thorin tmp]# tar -xzvf procps-3.1.13.tar.gz
2 procps-3.1.17/
3 procps-3.1.13/proc/
4 procps-3.1.13/proc/.cvsignore
5 .....
6 procps-3.1.13/watch.1
7 procps-3.1.13/watch.c
```

3. Przejdź do nowoutworzonego katalogu `./procps-3.1.13`. Komendą `ls` sprawdź zawartość katalogu.

Po rozpakowaniu pliku oprogramowania, w katalogu który został podczas rozpakowania utworzony należy rozejrzeć się za plikami o nazwach sugerujących, że należy zapoznać się z ich zawartością w pierwszej kolejności. Są to najczęściej: README, README.FIRST, INSTALL. W plikach tych została zawarta instrukcja jak należy postępować podczas instalacji programu. Czasami konieczne jest ustawienie jakiejś zmiennej środowiskowej ustalającej specyficzne opcje kompilatora, bez których instalacja skazana jest na niepowodzenie. W naszym przypadku odpowiednia sekcja wygląda następująco:

```
1 make distclean # clean-out everything to re-make from scratch
2 make           # takes about 0.75 minutes on a PPro 200 with SCSI
3 su             # for write/chown-perms on sys dirs
4 make install
5 ldconfig -v    # update ld.so to use new libproc if SHARED=1
6 exit
7 make distclean # remove anything that can be rebuilt
```

Tablica 1: Przykładowa sesja instalacji programu.

Należy zatem wykonywać kolejno komendy po stronie lewej znaczka `#`. Komenda `su` przełącza użytkownika na takiego, którego nazwa w systemie została podana jako argument wywołania komendy. Pominięcie argumentu oznacza przełączenie na użytkownika `root`. Komendę tą lepiej jest użyć z opcją `-`: `su -`, co spowoduje uaktualnienie środowiska zamiast pozostawienie odziedziczonego wprost po poprzednim użytkowniku. Należy wówczas pamiętać o przejściu do odpowiedniego katalogu, gdyż tak wywołana komenda przeniesie nas do katalogu osobistego nowego użytkownika. Jeśli jesteśmy podłączeni jako użytkownik `root`, to nie wykonujemy tej komendy.

Podczas instalacji mogą pojawić się błędy, np:

```

1 install --owner 0 --group 0 --mode a=r ps.1 //usr/man/man1/ps.1
2 install: cannot create regular file '//usr/man/man1/ps.1': No such file or directory
3 make[1]: *** [install] Error 1
4 make[1]: Leaving directory '/tmp/procps-2.0.7/ps'
5 make: *** [install_ps] Error 2

```

Oznacza to tyle, że nie udało się znaleźć katalogu `/usr/man/man1`, gdyż rozważana dystrybucja systemu przechowuje pliki dokumentacji w innym katalogu. Należy wówczas zajrzeć do pliku `/etc/man.config` i znaleźć odpowiedni katalog. Może to być np. `/usr/share/man`. Następnie edytujemy plik `Makefile`, odszukujemy w nim linię definiującą ścieżkę dostępu do manuala:

```

1 export MANDIR      = /usr/lib/man

```

i zmieniamy na poprawną:

```

1 export MANDIR      = /usr/share/man

```

Jeśli nie wywoływaliśmy komendy `su` (nie zmienialiśmy kontekstu użytkownika) to nie musimy również wywoływać komendy `exit` (patrz linia 6 tabeli 1).

Po zakończeniu instalacji przechodzimy do katalogu `/tmp` i usuwamy plik `procps-3.1.13.tar.gz` oraz katalog `procps-3.1.13`.

Zainstalowany pakiet umożliwia symboliczne zarządzanie procesami. Przykładowo, jeśli dotychczas chcieliśmy zwiększyć wartość parametru `NICE` wszystkich procesów dowolnego użytkownika, to wymagało to posłużenia się komendą `ps` dla odnalezienia numerów identyfikacyjnych wszystkich interesujących nas procesów, a następnie komendą `renice` dla ustawienia odpowiedniej wartości znalezionym procesom. W pakiecie `procps` znajduje się komenda `snice`, która jako argument wymaga podania nazwy użytkownika w systemie. Np:

```

1 [bory@thorin bory]$ snice +5 bory

```

spowoduje ustawienie wartości `NICE` o 5 większej od aktualnej wszystkim procesom użytkownika `bory`. W pakiecie znajdują się również polecenia `skill`, `pkill` oraz `pgrep`.

Instalacja oprogramowania w systemach z rodziny UNIX ma w większości przypadków znacznie bardziej uporządkowany charakter. Polega ona na dokonaniu instalacji z plików o specjalnym formacie z wykorzystaniem programów systemowych, które oprócz fizycznego umieszczenia plików instalowanego oprogramowania w odpowiednich katalogach dokonują ich weryfikacji, sprawdzają zgodność wersji oraz występowanie w systemie wymaganego, innego oprogramowania. Z wykorzystaniem zarządców oprogramowania możliwe jest również kompleksowe usuwanie oprogramowania wraz ze wszystkimi jego komponentami. Zarządzanie oprogramowaniem jest zazwyczaj prowadzone w oparciu o bazę danych.

W dystrybucji **RedHat** systemu *Linux* również istnieje zarządca pakietów zwany *Red Hat Package Manager* - `rpm`. Jest on stosowany przez inne dystrybucje jak np. *SuSE*. Może być używany do budowania, instalowania, zapytywania, weryfikowania, odświeżania i odinstalowywania pakietów oprogramowania. Pakiet składa się z archiwum plików oraz informacji pakietowej, takiej jak nazwa, wersja i opis.

Program `rpm` może pracować w jednym z następujących podstawowych trybów: inicjalizowania bazy danych, przebudowywania bazy danych, budowania pakietu, rekompilowania pakietu, budowania pakietu z archiwum `tar`, zapytywania, pokazywania tagów zapytania, instalowania, odświeżania, odinstalowywania, weryfikowania i innych.

Ogólna postać komendy instalowania `rpm` jest następująca:

```
rpm -i [opcje-instalowania] <plik_pakietu>
```

co zainstaluje nowy pakiet.

Ogólną postacią komendy odświeżania `rpm` jest:

```
rpm -U [opcje-instalowania] <plik_pakietu>
```

co odświeży lub zainstaluje pakiet na wersję, obecną w nowym RPM. Jest to to samo co opcja instalacji, lecz wszystkie inne wersje pakietu będą najpierw z systemu usuwane.

```
rpm [-F|-freshen] [opcje-instalowania] <plik_pakietu>
```

odświeży pakiety, lecz tylko jeśli wcześniejsza wersja już istnieje.

<plik_pakietu> może być podany jako ścieżka dostępu do pliku w systemie plików, jak również jako adres URL – ftp lub http. W tym wypadku pakiet zostanie pobrany przed zainstalowaniem z miejsca określonego adresem. Spośród opcji instalowania na uwagę zasługuje `-test`, która spowoduje sprawdzenie, czy nie istnieją konflikty bez instalowania pakietu.

Ogólną postacią komendy weryfikacji `rpm` jest

```
rpm -V|-y|-verify [opcje-weryfikacji] <plik_pakietu>
```

Weryfikowanie pakietu porównuje informacje o zainstalowanych plikach w pakiecie z informacją o plikach, pobraną z oryginalnego pakietu, zapisanego w bazie rpm. Wśród innych wartości, porównywane są rozmiary, sumy kontrolne MD5, prawa dostępu, typ, właściciel i grupa każdego pliku. Wszystkie niezgodności są natychmiast wyświetlane.

Ogólną postacią komendy odinstalowania rpm jest

```
rpm -e <nazwa_pakietu>
```

Spośród opcji na uwagę zasługują: `-allmatches` - usuń wszystkie wersje pakietu, które odpowiadają <nazwie_pakietu>, `-nodeps` - nie sprawdzaj zależności przed odinstalowaniem oraz `-test` - nie odinstalowuj niczego, przejdź tylko przez kolejne etapy procesu odinstalowywania.

4. Przejdź do katalogu `/tmp`. Pobierz z adresu `http://messy.icsr.agh.edu.pl/sysopy/` plik `sysstat-4.1.7-1.i386.rpm` zapisując go w bieżącym katalogu. Jest to plik rpm zawierający narzędzie służące pomiarom wydajności systemu.

5. Korzystając z komendy rpm zainstaluj skopiowany pakiet oprogramowania. Użyj opcji `-i` oraz `-v`, a jeśli pakiet jest już zainstalowany opcji `-F`.

Poprawna instalacja powinna przebiegać następująco:

```
1 [root@thorin tmp]# rpm -iv sysstat-4.1.7-1.i386.rpm
2 Preparing packages for installation...
3 sysstat-4.1.7-1
```

Nazwa `sysstat-4.1.7-1` z ostatniej linii posłuży do odinstalowania pakietu.

W pakiecie znajduje się program `sar` - *System Activity Report*, będący podstawowym narzędziem diagnostycznym w rodzinie AT&T Unix (SYSTEM V). Program posiada ponad 20 opcji (zainteresowanych odsyłam do manuala). Zazwyczaj wywołuje się go z dwoma argumentami. Pierwszy oznacza liczbę sekund między momentami próbkowania stanu systemu, drugi liczbę próbkowań (w przykładzie poniżej 5 próbkowań co 2 sekundy). Bez opcji `sar` podaje obciążenie procesorów systemu:

```
1 [root@thorin tmp]# sar 2 5
2 Linux 2.4.18-3 (thorin.icsr.agh.edu.pl)          2003-08-10
3
4 07:13:21      CPU      %user      %nice    %system    %idle
5 07:13:23      all       99,50        0,00        0,50        0,00
6 07:13:25      all      100,00        0,00        0,00        0,00
7 07:13:27      all       23,00        0,00        0,50       76,50
8 07:13:29      all        0,00        0,00        0,00      100,00
9 07:13:31      all        0,00        0,00        0,00      100,00
10 Average:      all       44,50        0,00        0,20       55,30
```

6. Korzystając z komendy `sar` zbadaj obciążenie systemu wejścia/wyjścia. Opcje znajdź w manualu.

7. Korzystając z komendy `rpm` odinstaluj pakiet `sysstat-4.1.7-1`.

Numeracja wersji jądra systemu Linux

Jądra systemu Linux są numerowane trzyczęściowym identyfikatorem: A.B.C, gdzie A,B i C są liczbami oznaczającymi kolejno: numer główny wersji (*major version number*), numer podwersji (*minor version number*) i numer rewizji (*revision version number*). Czasem może występować numer dodatkowy wersji. Stabilne wersje posiadają parzysty numer podwersji (drugi). Obecnie najnowsze dystrybucje oznaczone są numerem 2.6.x. Wersje te są polecane, jako wolne od dziur i błędów. Jądra o drugich numerach nieparzystych (2.1.x, 2.3.x, 2.5.x) są przeznaczone dla tych, którzy chcą je testować, wprowadzać jakieś zmiany, może nawet współpracować przy ich tworzeniu. Niestabilne jądra wykorzystują niektórzy zwariowani administratorzy i to jedynie w przypadkach, gdy potrzebują w swojej konfiguracji funkcjonalności niedostępnej w wersjach stabilnych. W wersjach tych („nieparzystych”) mogą być błędy, mogą się one zawieszać i prowadzić do utraty danych, chociaż to wcale nie musi być regułą. Aby dowiedzieć się o jakim numerze jądro jest zainstalowane na dowolnym komputerze pracującym pod kontrolą systemu operacyjnego Linux, należy wydać polecenie:

```
1 [bory@thorin bory]$ uname -r
2 2.4.18-3
```

Można również podejrzeć zawartość pliku `/proc/version`.

Systemy z rodziny UNIX wykorzystują rozwiązania, w których jądro w całości jest odpowiedzialne za obsługę wszystkich urządzeń „widzianych” przez system. W przypadku systemu *Linux*, jak również w innych nowych wersjach systemu UNIX, zrezygnowano z koncepcji jądra monolitu na rzecz bardziej elastycznej struktury. Poszczególne fragmenty jądra mogą być wydzielone w postaci modułów, które mogą być dynamicznie (podczas pracy systemu) dołączane. Koncepcja taka pozwala przygotować system do pracy w różnych konfiguracjach sprzętowych, wystarczy że w zależności od sprzętu wybierzemy i załadujemy potrzebne nam moduły.

Pomimo wydzielenia fragmentów jądra w postaci modułów nie zmienia to koncepcji budowy systemu gdyż moduły po załadowaniu stają się fragmentem jądra i posiadają te same możliwości co kod jądra. System nie monitoruje pracy modułu, nie ma nad nim kontroli i nie może zabronić mu wykonywania jakichkolwiek operacji, oznacza to również, że nie może stwierdzić, że moduł działa (nie)poprawnie. Modularność pozwala jedynie na rozdzielenie kodu i oszczędność pamięci.

Bez względu na rodzaj systemu UNIX podstawowym krokiem uruchomienia systemu jest załadowanie jądra systemu do pamięci i przekazanie mu sterowania. Sama operacja ładowania jądra systemu oraz kroki ją poprzedzające zależą bardzo od architektury komputera. Niezmienny jest natomiast fakt, że jądro systemu występuje gdzieś w pliku, który jest ładowany podczas startu. W przypadku systemu *Linux* uruchamianego z dysku twardego jądro systemu zazwyczaj jest umieszczone w katalogu `/boot` jako plik `vmlinuz-x.y.z`. Drugim elementem składowym jądra są moduły, które zazwyczaj umieszczone są w katalogu `/lib/modules/x.y.z`.

8. Sprawdź jaki jest rozmiar pliku zawierającego jądro systemu oraz podaj ile modułów jest dostępnych dla działającej obecnie wersji jądra

Ładowalne moduły są więc kawałkami kodu jądra, które nie znajdują się cały czas w pamięci, tak jak program jądra, a są tam ładowane tylko wtedy, gdy są potrzebne. Jeżeli, na przykład, skonfigurowano obsługę stacji dyskietek jako moduł, i postanowiono ją zamontować lub z niej czytać to specjalny demon sprawdzi czy sterownik tego urządzenia jest w pamięci i jeśli nie, to go załaduje. Teraz będzie można na tym urządzeniu wykonywać wszystkie operacje. Nieużywany sterownik może zostać usunięty z pamięci (faktycznie konieczne jest jeszcze spełnienie dodatkowych warunków, jak np. włączenie flagi *autoclean*). Takie rozwiązanie przy niewielkim narzucie czasowym oszczędza nam pamięć i pozwala obsługiwać wiele urządzeń, systemów plików itp. mimo niewielkiego rozmiaru jądra. Jest to niezwykle cenne w przypadku gdy system ma współpracować z bardzo różnymi konfiguracjami sprzętowymi, tak jak to ma miejsce w przypadku każdej dystrybucji systemu *Linux*. Dodatkową zaletą jest możliwość rekompilacji tylko modułów, a nie całego jądra w przypadku zmiany jakiegoś urządzenia w komputerze. Jeśli karta dźwiękowa obsługiwana jest sterownikiem - modułem, to przy jej wymianie na inną wystarczy zrekompilować tylko moduł jej obsługi i wymienić stary. Moduły można także ładować ręcznie za pomocą programu `insmod` i usuwać `rmmod`. Do zarządzania modułami służy również `modprobe`. Opcja `-a` powoduje dodanie modułu, opcja `-r` usuwa, opcja `-l` listuje bieżącą konfigurację. Zalecane jest stosowanie opcji `-v` wraz z opcjami `-r` oraz `-a`. Precyzyjnych informacji na temat konkretnego modułu dostarcza z kolei komenda `modinfo`, np: opcja `-n` wypisuje nazwy plików modułu, `-l` warunki licencji. Wiele interesujących szczegółów na ten temat można znaleźć w Module-HOWTO.

9. Załóżmy hipotetycznie, że w naszym systemie komputerowym na miejsce napędu CD zainstalowano nagrywarkę IDE. Konieczna jest zatem wymiana odpowiedniego modułu, a mianowicie moduł `ide-cd` musimy wymienić na moduł `ide-scsi`. Wykorzystując program `rmmod` usuń

moduł odpowiedzialny za obsługę urządzenia CD bezpośrednio poprzez interfejs ATA. Następnie korzystając z programu `modprobe` załaduj moduł `ide-scsi` który emuluje obsługę SCSI poprzez interfejs ATA.

Instalacja jądra systemu z wersji źródłowej

Zdobyte źródła (np. z przechowywanej u nas kopii (patrz punkt 11)), zazwyczaj jest to plik `linux-x.y.z.tar.gz` gdzie x, y, z są numerami wersji należy rozpakować do katalogu `/usr/src` wcześniej robiąc kopię zapasową poprzednich (jeśli istnieją):

```
1 # cd /usr/src
```

Sprawdź czy w tym katalogu jest katalog `linux`, jeśli tak, zmień jego nazwę np:

```
1 # mv linux linux-stary
```

Jeżeli znajduje się tu link symboliczny o nazwie `linux` to go usuń.

```
1 # rm -f linux
```

Teraz w katalogu `/usr/src` rozpakuj pliki zawierające kod źródłowy jądra poleceniem:

```
1 # tar xpvf linux-x.y.z.tar.gz
```

Po pomyślnym rozpakowaniu powinien pojawić się nowy katalog `linux`. Nie należy martwić się jeśli zamiast nazwy właściciela indywidualnego i grupowego katalogu `linux` pojawiają się cyferki. Wówczas zmieniamy właścicieli komendą:

```
1 #chown -R root:root linux
```

Jeśli archiwum nie zostało rozpakowane do katalogu o nazwie `linux-x.y.z` to dobrym pomysłem byłoby w tym momencie zmienić jego nazwę na `linux-x.y.z`. Teraz tworzymy symboliczne dołączenie do tego katalogu o nazwie `linux`. Aby to zrobić użyj polecenia:

```
1 # mv linux linux-x.y.z
2 # ln -s linux-x.y.z linux
```

Teraz zmień bieżący katalog na `linux` i przejrzyj plik `README`. - przeczytaj go, lub chociaż sprawdź, jaka wersja kompilatora `gcc` i biblioteki `libc` jest potrzebna do kompilacji. Sprawdź posiadaną wersję `gcc`:

```
1 $ gcc -v
```

Jej numer nie może być mniejszy od numeru podanego jako konieczny w pliku `README`.

Sprawdź czy istnieją linki symboliczne `/usr/include/asm`, `/usr/include/linux`, `/usr/include/scsi` do odpowiednich katalogów w źródłach i ewentualnie je utwórz:

Uwaga: W nowszych dystrybucjach wspomniane linki nie są niezbędne, gdyż potrzebne do kompilacji programów nagłówki zostały skopiowane do dystrybucji biblioteki `libc`.

```
1 # cd /usr/include
2 # rm -rf asm linux scsi
3 # ln -s /usr/src/linux/include/asm-i386 asm
4 # ln -s /usr/src/linux/include/linux linux
5 # ln -s /usr/src/linux/include/scsi scsi
```

W ogólnym przypadku należałoby zadbać o usunięcie starych plików *.o o ile takowe występują. W naszej sytuacji nie jest to zalecane gdyż, dostarczone przez nas jądro jest wstępnie skompilowane, co znacznie oszczędza czas potrzebny na kompilację.

```
1 # cd /usr/src/linux
2 # make mrproper
```

lub:

```
1 # cd /usr/src/linux
2 # make clean
```

Różnica między `make mrproper` a `make clean` jest taka, że ten pierwszy usuwa także poprzednie pliki z zapisaną konfiguracją. Jeśli istnieje konieczność kompilacji własnych, zmienionych źródeł, to zalecane jest wywołanie `make clean`, które jedynie sprawdzi konfigurację, a nie będzie konfigurować od początku.

Konfiguracja jądra

Przejdź do katalogu, w którym umieściłeś kod źródłowy nowego jądra i napisz:

```
1 # make config
```

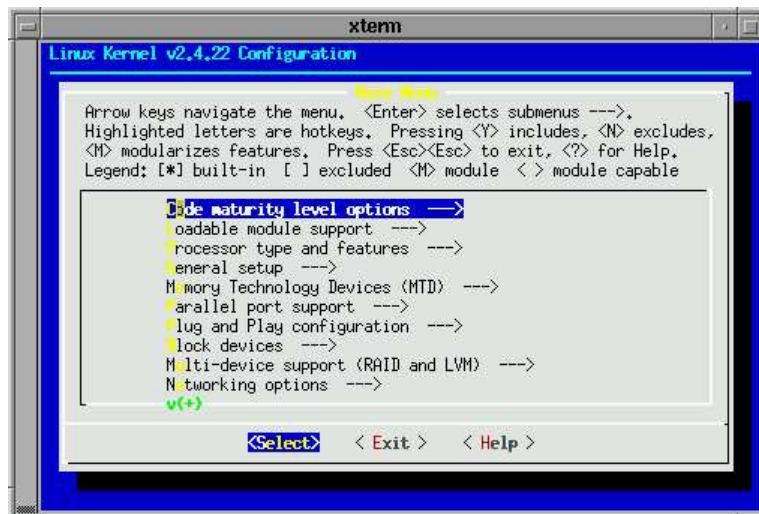
lub:

```
1 # make menuconfig
```

lub:

```
1 # make xconfig
```

`make config` jest prostym skryptem konfiguracyjnym, zadaje pytania na które należy odpowiadać Y=tak/ N=nie/ M=moduł = dla doświadczonych (nie zalecane przy pierwszej kompilacji jądra). Dla reszty przygotowano inny skrypt: `make menuconfig`. Są to przyjazne „okienka” w trybie tekstowym udostępniające podpowiedzi do każdej opcji. Dla znakomitej większości opcji udostępniany jest opis jak wpływa ona na zmianę działania jądra systemu oraz szczegóły konfiguracji sprzętu jeśli jest to na przykład sterownik. Można również użyć `make xconfig` - podobne do poprzedniego ale pod X Window (rysunek 1). Użycie któregośkolwiek z dwóch ostatnich sposobów konfiguracji jądra powoduje kompilację i uruchomienie aplikacji.



Rysunek 1: Menu główne konfiguratora z interfejsem X Window.

Kompilacja jądra

Kompilację nowego jądra rozpoczynamy od sprawdzenia i ustawienia zależności między plikami (*dependencies*). **Krok ten jest niezbędny po każdej zmianie konfiguracji jądra:**

```
1 # make dep
```

Najprostszym sposobem utworzenia jądra systemu jest wydanie polecenia

```
1 # make bzImage
```

powoduje to utworzenie programu – jądra systemu a następnie spreparowaniu takiej postaci, w której może być załadowane przez *loader*. Na etapie przygotowania jądra następuje kompresja pliku, dołączenie bieżącej konfiguracji oraz bloku startowego. Gotowa do użycia postać znajduje się w podkatalogu katalogu `arch/i386/boot`¹ jako plik o nazwie `zImage`. Kolejnym sposobem jest osiągnięcie celu `bzImage` za pomocą polecenia `make`. Różnica w porównaniu z wersją poprzednią polega na użyciu innego algorytmu kompresji skompilowanego kodu jądra, a ponadto jądro jest zapisywane do pliku o nazwie `bzImage`.

Utworzone w ten sposób jądro nadaje się do natychmiastowego użycia. Można je na przykład umieścić na dyskietce (o ile się zmieści) posługując się poleceniem `dd`.

Kompilacja jądra zajmuje dość dużo czasu, szczególnie wówczas gdy wybraliśmy wiele komponentów.

Jeśli na etapie konfiguracji zdecydowano się na używanie modułów i przynajmniej jeden z elementów jądra został wybrany jako moduł należy wykonać etap budowania modułów, czyli wydać polecenie:

```
1 # make modules
```

Instalacja skompilowanego jądra

Bez względu na sposób kompilacji oraz bez znaczenia czy robiliśmy to samodzielnie czy też otrzymaliśmy wersję skompilowaną przez kogoś innego musimy zachęcić komputer aby zaczął korzystać z nowej wersji.

Jądro systemu czyli plik `zImage` lub `bzImage` należy skopiować w miejsce gdzie znajdowała się wersja poprzednia czyli przypuszczalnie podkatalog `/boot`. Dla utrzymania porządku warto w podkatalogu `/boot` przechowywać obrazy jąder systemu pod nazwami zawierającymi numer wersji co ogranicza możliwość przypadkowego uszkodzenia poprzedniej wersji. Zwyczajowo są to pliki o nazwach `vmlinuz-x.y.z`.

W przypadku gdy ładowaniem naszego systemu zajmuje się program `LILO` edytujemy plik `/etc/lilo.conf`, dopisując do niego informacje o nowej lokalizacji jądra. Dla przykładu przyjmując, że plik nowego jądra to `vmlinuz-2.4.22` oraz, że korzeń systemu plików umieszczony jest na dysku `/dev/sda6` należy dopisać linie:

```
1 image=/boot/vmlinuz-2.4.22
2 label=przyklad
3 root=/dev/sda6
4 read-only
```

Dokładny opis polecenia `lilo` oraz pliku `lilo.conf` znajduje się w dokumentacji wbudowanej (`man`).

Po każdej zmianie dokonanej w pliku konfiguracyjnym `lilo.conf` lub na którymkolwiek z plików do których plik ten się odnosi, konieczne jest wykonanie polecenia `lilo`. Podczas wykonywania polecenie odczytywany i analizowany jest plik konfiguracyjny a następnie odczytywane są pliki opisane przez plik konfiguracyjny. Kolejnym etapem działania programu jest utworzenie pliku `map` – zazwyczaj umieszczonego w podkatalogu `/boot`, plik ten jest mapą pozwalającą na etapie uruchomienia loadera systemu odczytać pliki stanowiące jądro systemu. Proszę zwrócić uwagę, że loader podczas startu nie może posługiwać się systemem plików, jest zbyt mały aby pomieścić stosowne procedury, w związku z tym plik `map` zawiera numery wszystkich bloków które wchodzi w skład pliku jądra systemu co jest wystarczające jeśli zamierzamy odczytywać jedynie ten plik. Jeśli wszystko przebiegło poprawnie podczas uruchamiania komputera powinniśmy mieć możliwość wyboru nowego jądra, jednak do czasu przetestowania czy nowe jądro działa poprawnie warto pozostawić w konfiguracji programu `LILO` możliwość uruchomienia systemu z poprzedniej wersji.

Poza właściwym jądrem należy wykonać operację instalowania modułów jądra. W typowym przypadku gdy kompilujemy jądro dla potrzeb systemu na którym kompilacja jest wykonywana wystarczy wydanie polecenia:

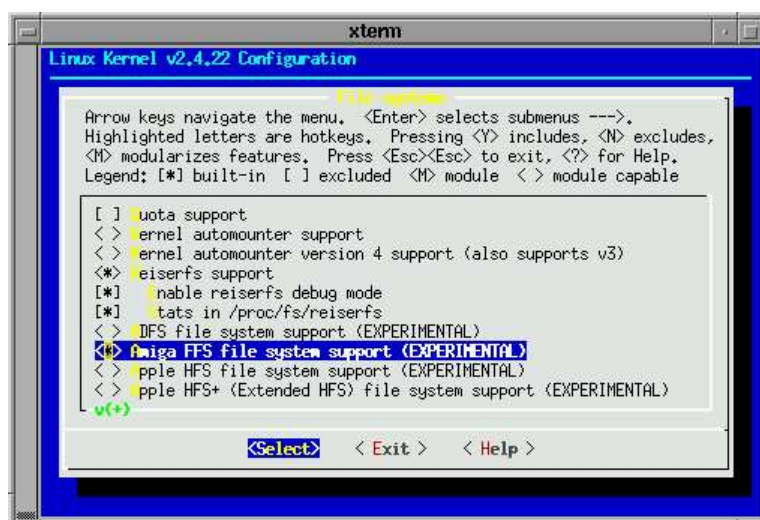
¹ w przypadku architektury `i386`

```
1 make modules_install
```

Podczas tworzenia tego celu wszystkie utworzone podczas kompilacji moduły zostaną skopiowane do katalogu odpowiedniego dla kompilowanej wersji jądra czyli `/lib/modules/x.y.z`. W przypadku gdy numer wersji kompilowanego jądra i któregoś z eksploatowanych jąder jest taki sam warto pamiętać o wykonaniu kopii katalogu z modułami oraz usunięciu wersji poprzedniej, gdyż pomimo zgodności wersji jądra załadowanie modułów dla różnych kompilacji nie zawsze jest możliwe.

Po skopiowaniu modułów do odpowiedniego katalogu (np. pośrednio przez program `make`) wymagane jest wyznaczenie zależności między poszczególnymi modułami, służy do tego polecenie `depmod`, które jest wykonywane zarówno podczas instalacji modułów jak również podczas uruchamiania systemu.

10. Korzystając z przytoczonego opisu kompilacji jądra utwórz własne jądro. Kod źródłowy pobierz z zasobu <http://messy.icsr.agh.edu.pl/sysopy/linux-2.4.22-sysop.tar.gz>. W nowym jądrze zapewnij obsługę następujących systemów plików: ext2, ext3, NTFS, NFS oraz reiserfs (rysunek 2). Zainstaluj również obsługę RAID oraz LVM. Nie poprawiaj pliku konfiguracyjnego `/etc/lilo.conf`. Sprawdź rozmiar nowego jądra.

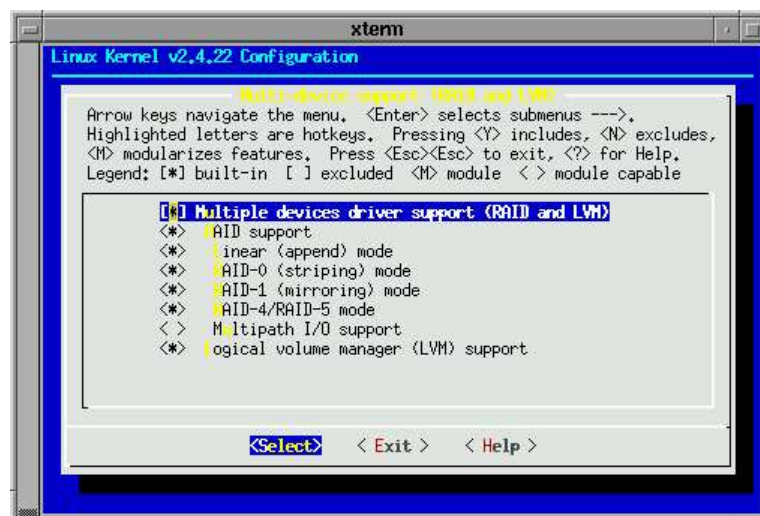


Rysunek 2: Menu systemu plików; dodajemy te wymagane w naszym systemie.

11. W trakcie kompilacji (co jest operacją czasochłonną) podłącz się raz jeszcze do systemu wykorzystując inną konsolę. Skopiuj plik <http://messy.icsr.agh.edu.pl/sysopy/lab4-kernel.tgz> do dowolnego katalogu i rozpakuj go. Archiwum zawiera jądro oraz katalog zawierający moduły. Przenieś je do stosownych katalogów. Zmodyfikuj plik konfiguracyjny `lilo.conf` tak aby było możliwe uruchomienie dostarczonego jądra poprzez etykietę `sysopy`.

12. Jeśli uruchomiona w punkcie 9 kompilacja została zakończona, to dołącz możliwość jego uruchamiania w pliku `lilo.conf` tworząc pozycję oznaczoną etykietą `nowe`.

13. Uruchom system w celu sprawdzenia, czy pracuje on poprawnie w każdej ze stworzonych konfiguracji jądra.



Rysunek 3: Menu służy dodaniu systemów RAID oraz LVM. Na zapas wybieramy wszystkie dostępne poziomy RAID.