7

Fajl-sistem */proc*

POKUŠAJTE POKRENUTI KOMANDU mount BEZ ARGUMENATA—to prikazuje fajl-sisteme trenutno podignute na vašem GNU/Linuks računaru. Videćete jednu liniju koja izgleda ovako:

```
none on /proc type proc (rw)
```

Ovo je specijalni fajl-sistem /proc. Primetite da prvo polje, none, indikuje da ovaj fajl-sistem nije povezan ni sa jednim hardverskim uređajem poput disk-jedinice. Umesto toga, /proc je prozor ka aktivnom Linuksovom jezgru. Fajlovi u /proc fajl-sistemu nemaju odgovarajuće fajlove na fizičkom uređaju; one su zapravo magični objekti koji se ponašaju kao fajlovi ali omogućavaju pristup parametrima, strukturama podataka, i statistikama unutar jezgra. "Sadržaj" ovih fajlova ne predstavlja uvek fiksne blokove podataka, kao što to čine obični fajlovi. Umesto toga, oni se generišu u letu od strane Linuksovog jezgra kada pokušate da čitate iz datog fajla. Možete takođe da promenite konfiguraciju tekućeg jezgra pisanjem u odgovarajuće fajlove u /proc fajl-sistemu. Pogledajmo primer:

```
% ls -l /proc/version
-r--r--r-- 1 root root 0 Jan 17 18:09 /proc/version
```

Obratite pažnju da je veličina fajla nula; pošto se sadržaj fajla generiše u letu, koncept veličine fajla ovde ne postoji. Takođe, ako sami pokrenete ovu komandu, primetićete da je vreme poslednje modifikacije fajla zapravo trenutno vreme.

Šta je u ovom fajlu? Sadržaj fajla /proc/version se sastoji od stringa koji opisuje verziju Linuksovog jezgra. Sadrži podatke o verziji koje bismo dobili sistemskim pozivom uname, opisanog u Poglavlju 8,“Linuksovi sistemski pozivi” u Sekciji 8.15, “uname,” plus dodatni podaci poput verzije kompajlera koji se koristio pri kompajliranju jezgra. Možete čitati iz fajla /proc/version kao što biste iz bilo kojeg drugog fajla. Na primer, jednostavan način da prikažete njegov sadržaj je pomoću komande cat.

```
% cat /proc/version
Linux version 2.2.14-5.0 (root@porky.devel.redhat.com) (gcc version egcs-2.91.
66 19990314/Linux (egcs-1.1.2 release)) #1 Tue Mar 7 21:07:39 EST 2000
```

Raznoliki zapisi u /proc fajl-sistemu su naširoko opisani u man-stranama za proc (Sekcija 5). Da biste je pogledali, pokrenite ovu komandu:

```
% man 5 proc
```

U ovom poglavlju, opisaćemo neke mogućnosti /proc fajl-sistema koje najverovatnije mogu biti korisne programerima aplikacija, i daćemo primere njihovog korištenja. Neke mogućnosti /proc fajl-sistema su zgodne za pronalaženje bagova, takođe.

Ako vas interesuje kako tačno radi /proc, pogledajte izvorni kod Linuksovog jezgra, pod /usr/src/linux/fs/proc/.

7.1 Vađenje podataka iz */proc-a*

Većina zapisa u /proc-u pružaju informacije prikazane na način čitljiv za ljude, ali su istovremeno dovoljno jednostavni da bi se lako parsirali. Na primer /proc/cpuinfo sadrži informacije o procesorskoj jedinici (ili procesorskim jedinicama, u slučaju višeprocesorskih mašina).

Prikaz sadrži tabelu vrednosti, po jednu u svakom redu, sa opisom vrednosti i dvotačkom pre svake vrednosti.

Na primer, prikaz može biti ovakav:

```
% cat /proc/cpuinfo
processor : 0
vendor_id : GenuineIntel
cpu family : 6
model : 5
model name : Pentium II (Deschutes)
stepping : 2
cpu MHz : 400.913520
cache size : 512 KB
fdiv_bug : no
hlt_bug : no
sep_bug : no
f00f_bug : no
coma_bug : no
fpu : yes
fpu_exception : yes
cpuid level : 2
wp : yes
flags : fpu vme de pse tsc msr pae mce cx8 apic sep
mtrr pge mca cmov pat pse36 mmx fxsr
bogomips : 399.77
```

Opisaćemo interpretaciju nekih od ovih polja u Sekciji 7.3.1, “Informacije o procesorkoj jedinici.”
Jednostavan način da se izvuče informacija je da se učita ceo fajl u memoriju a zatim iz nje koristeći funkciju sscanf. Listing 7.1 prikazuje primer ovoga. Program sadrži funkciju get_cpu_clock_speed koja čita iz /proc/cpuinfo u memoriju i učitava brzinu časovnika prve procesorske jedinice.

Listing 7.1 **(*clock-speed.c*) Dobavlja brzinu časovnika procesorske jedinice iz */proc/cpuinfo***

```
#include <stdio.h>
#include <string.h>
/* Vraća brzinu casovnika sistemskog procesora u MHz, kao sto pise u
/proc/cpuinfo. Na viseprocesorskih masinama, vraca brzinu prvog procesora.
Ako dodje do greske, vraca nulu. */

float get_cpu_clock_speed ()
{
  FILE* fp;
  char buffer[1024];
  size_t bytes_read;
  char* match;
  float clock_speed;

  /* Procitaj celokupan sadrzaj fajla /proc/cpuinfo u memorijski bafer. */
  fp = fopen ("/proc/cpuinfo", "r");
  bytes_read = fread (buffer, 1, sizeof (buffer), fp);
  fclose (fp);
  /* Izadji ako bafer nije dovoljno velik. */
  if (bytes_read == 0 || bytes_read == sizeof (buffer))
  return 0;
  /* Dodaj NUL-terminator na kraj stringa. */
  buffer[bytes_read] = '\0';
  /* Nadji liniju koja pocinje sa "cpu MHz". */
  match = strstr (buffer, "cpu MHz");
  if (match == NULL)
  return 0;
  /* Parsiraj red da bi izvukao brzinu casovnika. */
  sscanf (match, "cpu MHz : %f", &clock_speed);
  return clock_speed;
}

int main ()
{
  printf ("Brzina casovnika: %4.0f MHz\n", get_cpu_clock_speed ());
  return 0;
}
```

Pripazite, međutim, da imena, semantika, i format prikaza zapisa u fajl-sistemu /proc može biti predmet promene u novijim revizijama Linuksovog jezgra. Ako ih koristite u svom programu, trebali biste se pobrinuti da obezbedite "kulturno" ponašanje programa u slučaju da neki zapis iz /proc-a nedostaje ili je formatirana na neočekivan način.

7.2 Zapisi procesa

Fajl-sistem /proc sadrži direktorijum za svaki aktivan proces na GNU/Linuks sistemu. Ime svakog direktorijuma je identifikacioni broj odgovarajućeg procesa (PID)[1] Ovi direktorijumi se pojavljuju i nestaju dinamički u skladu sa nastajanjem i gašenjem procesa na sistemu. Svaki direktorijum sadrži nekoliko zapisa koji pružaju pristup informacijama o datom procesu. /proc fajl-sistem je zapravo dobio svoje ime po zapisima za ove procese.

Svaki direktorijum sadrži sledeće fajlove:

- cmdline sadrži listu argumenata komandne linije za ovaj proces. Fajl cmdline je opisana u Sekciji 7.2.2, “Lista argumenata procesa.”
- cwd je simbolički link koji vodi ka radnom direktorijumu procesa (podešenog, na primer, pomoću poziva chdir).
- environ sadrži okruženje procesa. Fajl environ je opisana u Sekciji 7.2.3, “Okruženje procesa.”
- exe je simbolički link koji pokazuje na izvršni fajl koji je pokrenuo proces. Fajl exe je opisana u Sekciji 7.2.4, “Izvršni fajl procesa.”
- fd je poddirektorijum koji sadrži zapise za fajl-deskriptore otvorene od strane procesa. Ovi su opisani u Sekciji 7.2.5, "Fajl-deskriptori procesa".
- maps prikazuje informacije o fajlovima mapiranim od strane procesa. Pogledajte poglavlje 5, "Međuprocesna komunikacija", Sekciju 5.3 "Mapirana memorija" radi detalja o tome kako funkcionišu memorijski mapirani fajlovi. Za svaki mapirani fajl, maps prikazuje opseg od koje do koje adrese je mapiran fajl u okviru memorijskog prostora datog procesa, dozvole nad ovim adresama, ime fajla, i ostale informacije. Tabela maps za svaki proces prikazuje izvršni fajl procesa, kakve god postoje učitene biblioteke da postoje, i ostale fajlove koje je proces mapirao.
- root je simbolički link na koreni direktorijum procesa. Ovo je obično link ka /, sistemskom korenom direktorijumu. Koreni direktorijum procesa se može promeniti sistemskim pozivom chroot ili komandom chroot[2].

---

[1] Na nekim UNIX sistemima, identifikacioni brojevi procesa su zapisani sa vodećim nulama. Na GNU/Linuksu, ovo nije slučaj.

[2] Poziv i komanda chroot su van dosega ove knjige. Pogledajte man-stranice za chroot u Sekciji 1 radi informacija o komandi (pokrenite man 1 chroot), ili man-stranicu za sistemski poziv chroot u Sekciji 2 (pokrenite man 2 chroot).

§ stat sadrži mnogo statusnih i statističkih podataka o procesu. Ovi su isti kao i oni u zapisu status, ali u sirovom numeričkom formatu, svi u jednom redu. Format je težak za čitanje ali verovatno zgodniji za učitavanje iz programa. Ako želite da koristite stat u svojim programima, pogledajte man-stranicu za proc,
koja opisuje sadržaj istog, tako što ćete pokrenuti komandu man 5 proc.
§ statm sadrži informacije o memoriji koju koristi proces. Zapis statm
je opisan u sekciji 7.2.6, "Statistike memorije procesa."
§ status sadrži mnogo statusnih i statističkih informacija o procesu, formatiranih tako da ih ljudi mogu lako razumeti. Sekcija 7.2.7, "Statistike procesa" sadrži opis zapisa status.
Zapis cpu se javlja samo kod SMP Linuksovih jezgara. Sadrži razlomljene podatke o procesorskim vremenima (korisničkim i sistemskim).
Obratite pažnju da zbog sigurnosnih razloga, dozvole nad nekim zapisima su podešene tako da samo vlasnik procesa (ili administrator) može da im pristupi.

7.2.1 */proc/self*

Jedan dodatni zapis u /proc fajl-sistemu nam olakšava korištenje /proc-a
u pronalaženju sopstvenog procesa. Zapis /proc/self je simbolički link ka direktorijumu /proc-a koji odgovara tekućem procesu. Odredište linka /proc/self zavisi od toga koji proces gleda u njega: Svaki proces vidi svoj radni direktorijum kao odredište linka.
Na primer, Listing 7.2 prikazuje program koji čita odredište linka /proc/self da bi odredio PID samog sebe. (Radimo to na ovaj način radi ilustracije; poziv funkcije getpid, opisane u Poglavlju 3, "Procesi", u Sekciji 3.1.1, "Identifikacioni brojevi procesa," je mnogo lakši način da se postigne ista stvar.). Ovaj program koristi sistemski poziv readlink, opisan u Sekciji 8.11, "readlink: čitanje simboličkih linkova," da bi pronašao odredište simboličkog linka.

Listing 7.2 **(*get-pid.c*) Dobavlja PID iz */proc/self***

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>

/* Vraca PID pozivajuceg procesa, u skladu sa simbolickim linkom /proc/self. */

pid_t get_pid_from_proc_self ()
{
  char target[32];
  int pid;
  /* Procitaj odrediste simbolicnog linka. */
  readlink ("/proc/self", target, sizeof (target));
```

nastavak na narednoj strani

Listing 7.2 nastavak sa prethodne strane

```
    /* odrediste je direktorijum sa imenom po identifikacionom broju procesa */
    sscanf (target, "%d", &pid);
    return (pid_t) pid;
}

int main ()
{
    printf ("/proc/self prijavljuje PID %d\n", (int) get_pid_from_proc_self ());
    printf ("getpid() prijavljuje PID %d\n", (int) getpid ());
    return 0;
}
```

7.2.2 Lista argumenata procesa

Zapis cmdline sadrži listu argumenata procesa (vidi Poglavlje 2, "Pisanje dobrog GNU/Linuks softvera", Sekciju 2.1.1, "Lista argumenata"). Argumenti su predstavljeni u obliku jedinstvenog niza karaktera, svaki razdvojen NUL-terminatorom od drugog. Većina funkcija za stringove očekuje da su stringovi završeni NUL-terminatorom, pa neće podržati ugneždene NUL-terminatore, tako da ćete morati da rukujete ovim sadržajem na neki drugi način.

NUL vs. NULL
NUL je specijalni karakter sa brojnom vrednošću 0. Razlikuje se od NULL, koji predstavlja pokazivač sa vrednošću 0. U C-u, niz karaktera se obično završava NUL karakterom (NUL-terminatorom). Na primer, niz karaktera "Zdravo, svete!" zauzima 14 bajtova jer postoji implicitno ubačen NUL nakon znaka uzvika, koji označava kraj stringa.
NULL je, s druge strane, pokazivač za koji uvek možete biti sigurni da mu ne odgovara nijedna stvarna memorijska lokacija u programu.
U programskim jezicima C i C++, NUL se izražava literalom '\0', ili kao (char) 0. Definicija konstante NULL se razlikuje od sistema do sistema. U Linuksu, definiše se kao ((void*)0) u C-u ili jednostavno 0 u C++-u.

U Sekciji 2.1.1, predstavili smo program u Listingu 2.1 koji je štampao sopstvenu listu argumenata. Koristeći zapise cmdline u fajl-sistemu /proc, možemo implementirati program koji štampa listu argumenata nekog drugog procesa. Listing 7.3 je jedan takav program; on štampa listu argumenata procesa sa zadatim PID-om. Pošto može biti nekoliko NUL-terminatora unutar sadržaja fajla cmdline a ne samo jedan na kraju, onda ne možemo odrediti dužinu stringa sa strlen (koja jednostavno broji karaktere jedan po jedan dok ne naiđe na NUL). Umesto toga, određujemo dužinu sadržaja pozivom funkcije read, koja vraća broj učitanih karaktera.

Listing 7.3 **(*print-arg-list.c*) Štampa listu argumenata tekućeg procesa**

```
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>

/* Stampa listu argumenata, argument po liniji, procesa sa zadatim PID-om */

void print_process_arg_list (pid_t pid)
{
    int fd;
    char filename[24];
    char arg_list[1024];
    size_t length;
    char* next_arg;

    /* Generisi ime potrebnog fajla cmdline za dati proces. */
    snprintf (filename, sizeof (filename), "/proc/%d/cmdline", (int) pid);
    /* Procitaj sadrzaj fajla. */
    fd = open (filename, O_RDONLY);
    length = read (fd, arg_list, sizeof (arg_list));
    close (fd);
    /* read ne terminira string, tako da cemo to uraditi rucno. */
    arg_list[length] = '\0';

    /* Prodji kroz argumente. Argumenti su razdvojeni NUL-terminatorom. */
    next_arg = arg_list;
    while (next_arg < arg_list + length) {
        /* Stampaj argument. Svaki je NUL-terminiran, pa ga tretiraj kao obicni string. */
        printf ("%s\n", next_arg);
        /* Predji na naredni argument. Posto je svaki NUL-terminiran, strlen meri duzinu
        narednog, ne citave liste argumenata. */
        next_arg += strlen (next_arg) + 1;
    }
}

int main (int argc, char* argv[])
{
    pid_t pid = (pid_t) atoi (argv[1]);
    print_process_arg_list (pid);
    return 0;
}
```

Na primer, pretpostavimo da je proces 372 sistemski demon za logovanje, syslogd.

```
% ps 372
PID TTY STAT TIME COMMAND
372 ? S 0:00 syslogd -m 0
% ./print-arg-list 372
syslogd
-m
0
```

U ovom slučaju, syslogd je bio pozvan sa argumentima -m 0.

7.2.3 Okruženje procesa

Zapis environ sadrži okruženje procesa (pogledajte sekciju 2.1.6, "Okruženje"). Kao i kod cmdline, zasebne promenjive okruženja su razdvojene NUL-terminatorima. Format svakog elementa je isti kao i u promenjivoj environ,
dakle PROMENJIVA=vrednost.
Listing 7.4 predstavlja generalizaciju programa u Listingu 2.3, u Sekciji 2.1.6.
Ova verzija uzima PID na komandnoj liniji i štampa okruženje za odgovarajući proces tako što ga čita iz /proc.

Listing 7.4 **(*print-environment.c*) Prikazuje okruženje procesa**

```
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>

/* Stampa okruzenje, jednu promenjivu po liniji, procesa zadatog preko PID-a. */

void print_process_environment (pid_t pid)
{
    int fd;
    char filename[24];
    char environment[8192];
    size_t length;
    char* next_var;

    /* Generisi ime fajla environ za zadati proces. */
    snprintf (filename, sizeof (filename), "/proc/%d/environ", (int) pid);
    /* Citaj sadrzaj tog fajla. */
    fd = open (filename, O_RDONLY);
    length = read (fd, environment, sizeof (environment));
    close (fd);
    /* read ne stavlja automatski NUL-terminator, pa mi to radimo rucno. */
    environment[length] = '\0';
```

```
    /* Prodji kroz promenjive. Promenjive su razdvojene znacima NUL. */
    next_var = environment;
    while (next_var < environment + length) {
        /* Stampaj promenjivu. Svaka je NUL-terminirana, pa ih gledamo kao
        obicne stringove. */
        printf ("%s\n", next_var);
        /* Pomeri se na narednu promenjivu. Posto je svaka promenjiva
        NUL-terminirana, strlen meri duzinu naredne promenjive, a ne citave
        liste promenjivih okruzenja. */
        next_var += strlen (next_var) + 1;
    }
}

int main (int argc, char* argv[])
{
    pid_t pid = (pid_t) atoi (argv[1]);
    print_process_environment (pid);
    return 0;
}
```

7.2.4 Izvršni fajl procesa

exe zapis pokazuje na izvršni fajl koji predstavlja proces. U Sekciji 2.1.1, objasnili smo da se ime izvršnog fajla obično prosledi kao prvi element liste argumenata. Obratite pažnju, međutim, da je ovo samo stvar konvencije; program može biti pokrenut sa bilo kojom listom argumenata. Koristeći zapis exe u fajl-sistemu /proc je sigurniji način da se odredi koji izvršni fajl je pokrenut. Jedna korisna tehnika je dobavljanje staze direktorijuma koji sadrži izvršni fajl, preko /proc fajl-sistema. Mnogi programi svoje pomoćne fajlove smeštaju u direktorijume čije se staze pamte relativno u odnosu na direktorijum gde se nalazi izvršni fajl, tako da je neophodno odrediti gde se on nalazi. Funkcija get_executable_path u Listingu 7.5 određuje stazu izvršnog fajla koji je pokrenut u tekućem procesu, ispitujući simbolički link /proc/self/exe.

Listing 7.5 **(*get-exe-path.c*) Pronalazi stazu do programa tekućeg procesa**

```
#include <limits.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>

/* Pronalazi stazu koja sadrzi izvrsni fajl tekuceg procesa. Ova staza se smesta u BUFFER,
koji je duzine LEN. Vraca broj karaktera u stazi, ili -1 ako dodje do greske. */
```

nastavak na narednoj strani

Listing 7.5 ***nastavak sa prethodne strane***

```
size_t get_executable_path (char* buffer, size_t len)
{
   char* path_end;
   /* Procitaj odrediste linka /proc/self/exe. */
   if (readlink ("/proc/self/exe", buffer, len) <= 0)
   return -1;
   /* Nadji poslednju pojavu kose crte, koja razdvaja imena direktorijuma u stazi. */
   path_end = strrchr (buffer, '/');
   if (path_end == NULL)
   return -1;
   /* Predji na karakter nakon kose crte. */
   ++path_end;
   /* Dobavi direktorijum programa odsecajuci deo nakon poslednje kose crte. */
   *path_end = '\0';
   /* Duzina staze je broj karaktera do poslednje kose crte. */
   return (size_t) (path_end - buffer);
}

int main ()
{
   char path[PATH_MAX];
   get_executable_path (path, sizeof (path));
   printf ("Ovaj program se nalazi u direktorijumu %s\n", path);
   return 0;
}
```

7.2.5 Fajl-deskriptori procesa

Zapis `fd` je poddirektorijum koji sadrži zapise po jedan za svaki fajl-deskriptor otvoren u procesu. Svaki zapis je simbolički link na fajl ili uređaj otvoren za taj fajl-deskriptor. Možete pisati u ili čitati iz ovih simboličkih linkova; ovo će predstavljati pisanje odnosno čitanje iz odgovarajućeg fajla ili uređaja otvorenog u datom procesu. Zapisi u poddirektorijumu `fd` imaju imena kao brojevi odgovarajućih fajl-deskriptora.

Evo simpatičnog trika kojeg možete da probate sa zapisima u direktorijumu `fd` fajl-sistema `/proc`. Otvorite novu konzolu, i nađite PID procesa ljuske pokretanjem komande `ps`.

```
% ps
PID TTY TIME CMD
1261 pts/4 00:00:00 bash
2455 pts/4 00:00:00 ps
```

U ovom slučaju, ljuska (bash) ima identifikacioni broj procesa 1261. Sad otvorite novu konzolu u drugom prozoru, i pogledajte sadržaj poddirektorijuma fd za taj proces.

```
% ls -l /proc/1261/fd
total 0
lrwx------ 1 samuel samuel 64 Jan 30 01:02 0 -> /dev/pts/4
lrwx------ 1 samuel samuel 64 Jan 30 01:02 1 -> /dev/pts/4
lrwx------ 1 samuel samuel 64 Jan 30 01:02 2 -> /dev/pts/4
```

(Može biti i dodatnih redova za neke druge otvorene fajl-deskriptore) Prisetite se šta smo rekli u Sekciji 2.1.4, "Standardni U/I", da se fajl-deskriptori 0, 1, i 2 inicijalno postavljaju da predstavljaju standardni ulazi, izlaz i izlaz za greške, tim redom. Prema tome, pisanjem u fajl /proc/1261/fd/1, vi ćete zapravo pisati na uređaj koji je povezan na stdout za ljuskin proces—u ovom slučaju na pseudo-terminal u prvom prozoru kojeg ste otvorili. Probajte da otkucate poruku u taj fajl, u drugom otvorenom prozoru::

```
% echo "Zdravo, svete!" >> /proc/1261/fd/1
```

Tekst se pojavljuje u prvom prozoru.

Osim fajl-deskriptora standardnog ulaza, izlaza i izlaza za greške se takođe pojavljuju u poddirektorijumu fd. Listing 7.6 prikazuje program koji jednostavno otvara fajl-deskriptor na fajl koji se zada kao argument komandne linije, a zatim ulazi u beskonačnu petlju.

Listing 7.6 **(*open-and-spin.c*) Otvara fajl za čitanje**

```
#include <fcntl.h>
#include <stdio.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>

int main (int argc, char* argv[])
{
   const char* const filename = argv[1];
   int fd = open (filename, O_RDONLY);
   printf ("u procesu %d, fajl-deskriptor %d je povezan sa fajlom %s\n",
   (int) getpid (), (int) fd, filename);
   while (1);
   return 0;
}
```

Pokušajte pokrenuti ovaj program u jednom prozoru:

```
% ./open-and-spin /etc/fstab
u procesu 2570, fajl-deskriptor 3 je povezan sa fajlom /etc/fstab
```

U drugom prozoru, pogledajte poddirektorijum fd koji odgovara ovom procesu u fajl-sistemu /proc.

```
% ls -l /proc/2570/fd
total 0
lrwx------ 1 samuel samuel 64 Jan 30 01:30 0 -> /dev/pts/2
```

```
lrwx------ 1 samuel samuel 64 Jan 30 01:30 1 -> /dev/pts/2
lrwx------ 1 samuel samuel 64 Jan 30 01:30 2 -> /dev/pts/2
lr-x------ 1 samuel samuel 64 Jan 30 01:30 3 -> /etc/fstab
```

Primetite da je zapis za fajl-deskriptor 3, zapravo link na fajl /etc/fstab koji je otvoren na ovom deskriptoru.

Fajl-deskriptori mogu biti otvoreni i za sokete i pajpove, takođe (pogledajte Poglavlje 5 radi više informacija o istim). U tom slučaju, odredište simboličnog linka za odgovarajući fajl-deskriptor će biti prikazano kao “socket” ili “pipe” umesto na neki obični fajl ili uređaj.

7.2.6 Statistike memorije procesa

Zapis statm sadrži listu od sedam brojeva, razdvojenih prazninama. Svaki broj je broj stranica memorije korištene od strane datog procesa u određenoj kategoriji. Kategorije, u redosledu pojavljivanja brojeva, su navedeni ovde:

- Ukupna veličina procesa
- Veličina procesa rezidentna u fizičkoj memoriji
- Memorija deljena sa drugim procesima—tj. memorija mapirana i od ovog procesa i od barem još nekog procesa (kao npr. deljene biblioteke ili neizmenjene stranice po principu "kopiraj-po-pisanju" (eng. copy-on-write))
- Veličina teksta procesa—tj. veličina učitanog izvršnog fajla
- Veličina deljenih biblioteka mapiranih u ovom procesu
- Memorija procesa korištena za stek.
- Broj "prljavih" stranica—tj. stranica koje su izmenjene od strane procesa.

7.2.7 Statistike procesa

Zapis status sadrži mnoštvo informacija o procesu, formatiranih da ih ljudi mogu lako razumeti. Među ovim informacijama je identifikacioni broj procesa i identifikacioni broj roditeljskog procesa, realni i efektivni korisnički i grupni ID, količina upotrebljene memorije, i bit-maske koje određuju koji signali se hvataju, koji ignorišu a koji blokiraju.

7.3 Informacije o hardveru

Nekoliko od preostalih zapisa u fajl-sistemu /proc obezbeđuju prilaz informacijama o sistemskom hardveru. Iako se za ove obično zanimaju sistemski konfiguratori i administratori, one mogu biti ponekad od koristi i programerima aplikacija. Predstavićemo neke od ovih zapisa ovde.

7.3.1 Informacije o procesorkoj jedinici

Kao što smo videli ranije, /proc/cpuinfo sadrži informacije o procesorskoj jedinici ili jedinicama koje su aktivne na sistemu. Polje Processor prikazuje broj procesora; ovo je 0 za jednoprocesorske sisteme. Polja Vendor, CPU Family, Model, i Stepping vam omogućavaju da odredite tačan model i reviziju procesora. I korisnije od toga, polje Flags prikazuje koje zastavice procesora su uključene, što govori o mogućnostima datog procesora. Na primer, "mmx" indikuje dostupnost proširenih MMX instrukcija[3]. Većina informacija vraćenih iz /proc/cpuinfo je dobijena iz x86 asemblerske instrukcije cpuid. Ova instrukcija predstavlja mehanizam niskog nivoa pomoću koje program dobavlja informacije o procesorskoj jedinici. Radi boljeg razumevanja sadržaja /proc/cpuinfo, pogledajte dokumentaciju o instrukciji cpuid u Intelovoj knjizi *IA-32 Uputstvo o Intel arhitekturi za programere, Tom 2: Spisak instrukcija.* Ovo uputstvo se može preuzeti sa adrese http://developer.intel.com/design.

Poslednji element, bogomips, je vrednost specifična za Linuks. On predstavlja meru brzine procesora dok se kreće u uskoj petlji i stoga predstavlja prilično slab indikator ukupne brzine procesora.

7.3.2 Informacije o uređajima

Fajl /proc/devices nam daje listing *glavnih brojeva* karakter- i blok-uređaja dostupnih sistemu. Pogledajte poglavlje 6, "Uređaji," radi više informacija o tipovima uređaja i brojeva uređaja.

7.3.3 Informacije o PCI magistrali

Fajl /proc/pci daje pregled uređaja prikačenih na PCI magistralu ili magistrale. To su konkretne PCI kartice, i mogu da uključuju uređaje integrisane u matičnu ploču, plus AGP grafičke karte. Pregled uključuje tip uređaja; ID uređaja i proizvođača; ime uređaja, ako postoji; informacije o mogućnostima datog uređaja; i informacije o PCI resursima koje koristi taj uređaj.

7.3.4 Informacije o serijskom portu

Fajl /proc/tty/driver/serial prikazuje informacije i statistike konfiguracije serijskih portova. Serijski brojevi se numerišu od nule (0)[4]. Informacije o konfiguraciji serijskih portova se takođe može nabaviti, kao i promeniti, koristeći komandu setserial. Međutim, /proc/tty/driver/serial prikazuje dodatne statistike o broju prekida svakog serijskog porta.

---

[3] Pogledajte *IA-32 Uputstvo o Intel arhitekturi za programere* za dokumentaciju o MMX instrukcijama, i pogledajte Poglavlje 9, "Ugnežden asemblerski kod" u ovoj knjizi, radi informacija o tome kako koristiti ove i druge specijalne asemblerske instrukcije u GNU/Linuks programima.

[4] Obratite pažnju da se serijski portovi pod DOS-om i Windows-om numerišu od broja 1, tako da COM1 odgovara serijskom portu 0 pod Linuksom.

Na primer, ovaj red teksta iz /proc/tty/driver/serial bi mogao opisivati serijski port 1 (što bi bilo COM2 pod Windows-om):

```
1: uart:16550A port:2F8 irq:3 baud:9600 tx:11 rx:0
```

Ovo nam govori da UART tipa 16550A pokreće serijski port, da ovaj koristi U/I port 0x2f8
i IRQ 3 za komunikaciju, i teče na 9,600 baud. Ovaj serijski port je je "video" 11 prekida za prenos i 0 prekida za prijem.
Pogledajte Sekciju 6.4, “Hardverski uređaji,” radi informacija o serijskim uređajima.

7.4 Informacije o jezgru

Mnogi zapisi u /proc nam daju pristup informacijama o stanju i konfiguraciji tekućeg jezgra. Neki od ovih zapisa su na prvom nivou direktorijuma /proc; neki se nalaze na stazi /proc/sys/kernel.

7.4.1 Informacije o verziji

Fajl /proc/version sadrži dugačak string koji opisuje verziju jezgra i verziju kreiranja jezgra. Takođe sadrži informacije o tome kako je jezgro kreirano: korisnika koji ga je kompajlirao, mašinu na kojoj je bilo kompajlirano, datum kad je bilo kompajlirano i verziju kompajlera koji je bio korišten—na primer:

```
% cat /proc/version
Linux version 2.2.14-5.0 (root@porky.devel.redhat.com) (gcc version
egcs-2.91.66 19990314/Linux (egcs-1.1.2 release)) #1 Tue Mar 7
21:07:39 EST 2000
```

Ovo nam govori da je sistem na jezgru verzije 2.2.14, koji je kompajliran EGCS-om verzije 1.1.2 (EGCS, eng. *Experimental GNU Compiler System - Eksperimentalni GNU kompajler sistem*, je predstavljao prethodnicu današnjeg projekta GCC.)
Najbitniji elementi u ovom prikazu, ime operativnog sistema i verzija i revizija jezgra, su dostupni u posebnim zapisima /proc-a. To su /proc/sys/, kernel/ostype, /proc/sys/kernel/osrelease, i /proc/sys/kernel/version, tim redom.

```
% cat /proc/sys/kernel/ostype
Linux
% cat /proc/sys/kernel/osrelease
2.2.14-5.0
% cat /proc/sys/kernel/version
#1 Tue Mar 7 21:07:39 EST 2000
```

7.4.2 Ime računara i ime domena

Fajlovi /proc/sys/kernel/hostname i /proc/sys/kernel/domainname sadrže ime računara i ime domena, tim redom. Ovi podaci su isti oni koji se vrate sistemskim pozivom uname, opisanom u Sekciji 8.15.

7.4.3 Upotreba memorije

Zapis /proc/meminfo sadrži informacije o količini upotrebljene memorije na sistemu. Informacije su predstavljene i za fizičku i za virtuelnu memoriju. Prva tri reda predstavljaju ukupne količine, u bajtovima. Naredne linije sumiraju ove informacije u kilobajtima—na primer:

```
% cat /proc/meminfo
        total:    used:    free:  shared : buffers:  cached:
Mem: 529694720 519610368 10084352 82612224 10977280 82108416
Swap: 271392768 44003328 227389440
MemTotal: 517280 kB
MemFree: 9848 kB
MemShared: 80676 kB
Buffers: 10720 kB
Cached: 80184 kB
BigTotal: 0 kB
BigFree: 0 kB
SwapTotal: 265032 kB
SwapFree: 222060 kB
```

Ovo prikazuje 512MB fizičke memorije, od koje je 9MB slobodno, i 258MB virtuelne memorije, od koje je 216MB slobodno. U liniji za fizičku memoriju, još tri vrednosti su prisutne:

- Kolona Shared predstavlja ukupnu količinu deljene memorije trenutno alocirane u sistemu (Pogledajte Sekciju 5.1, “Deljena memorija”).
- Kolona Buffers prikazuje memoriju alociranu od strane Linuksa za bafere blok-uređaja. Ove bafere koriste drajveri uređaja da bi čuvali podatke koji se upisuju na uređaj ili čitaju sa uređaja.
- Kolona Cached prikazuje memoriju koju je Linuks alocirao za keš straničenja. Ova memorija se koristi za keširanje pristupa mapiranim fajlovima.

Možete koristiti komandu free da dobijete sve ove iste informacije o memoriji.

7.5 Uređaji, podizanja (mount) i fajl-sistemi

Fajl-sistem /proc takođe sadrži informacije o diskovnim uređajima i fajl-sistemima podignutim sa njih (mount).

7.5.1 Fajl-sistemi

Zapis /proc/filesystems prikazuje tipove fajl-sistema koje jezgro poznaje. Primetite da lista nije mnogo krisna jer nije kompletna: fajl-sistemi mogu biti učitavani i iščitavani dinamički kao moduli jezgra. Sadržaj fajla /proc/filesystems izlistava samo tipove fajl-sistema koji su ili statički povezani u jezgro ili su trenutno učitani. I drugi tipovi fajl-sistema mogu biti dostupni sistemu kao moduli ali možda još nisu učitani.

7.5.2 Uređaji i particije

Fajl-sistem /proc uključuje informacije o uređajima spojenim i na IDE kontrolere i na SCSI kontrolere (ako ih ima na sistemu). Na tipičnim sistemima, poddirektorijum /proc/ide može da sadrži ili poddirektorijum ide0 ili ide1, ili oba, koji odgovaraju primarnom i sekundarnom IDE kontroleru na sistemu.[5] Ovi sadrže još poddirektorijuma koji odgovaraju fizičkim uređajima prikačenim na kontrolere. Direktorijumi za kontrolere i uređaje možda ne budu prisutni ako Linuks nije prepoznao nijedan spojeni uređaj. Pune staze do direktorijuma koji odgovaraju četiri moguća IDE uređaja su prikazane u Tabeli 7.1.

Tabela 7.1 **Pune staze direktorijuma koji odgovaraju četiri moguća IDE uređaja**

Controller	Device	Subdirectory
Primary	Master	/proc/ide/ide0/hda/
Primary	Slave	/proc/ide/ide0/hdb/
Secondary	Master	/proc/ide/ide1/hdc/
Secondary	Slave	/proc/ide/ide1/hdd/

Pogledajte Sekciju 6.4, "Hardverski uređaji", radi više informacija o imenima IDE uređaja. Svaki direktorijum za IDE uređaj pruža po nekoliko zapisa koji omogućavaju pristup informacijama o identifikaciji i konfiguraciji uređaja. Nekoliko najkorisnijih je navedeno ovde:

- model sadrži identifikaciju modela uređaja.
- media sadrži tip medijuma. Moguće vrednosti su disk, cdrom, tape, floppy, i UNKNOWN.
- capacity sadrži kapacitet uređaja, u 512-bajtnim blokovima. Obratite pažnju da će vrednost za CDROM uređaje biti 2^{31}–1, a ne kapacitet diska u uređaju. Obratite pažnju da capacity predstavlja kapacitet celog fizičkog diska; kapacitet fajl-sistema sadržanih u particijama će biti manji.

Na primer, ove komande nam prikazuju kako da odredimo tip medijuma i identifikaciju uređaja za master uređaj na sekundarnom IDE kontroleru. U ovom slučaju, izlazi da je u pitanju Toshiba CD-ROM drive.

```
% cat /proc/ide/ide1/hdc/media
cdrom
% cat /proc/ide/ide1/hdc/model
TOSHIBA CD-ROM XM-6702B
```

[5] Ako se pravilno konfiguriše, Linuksovo jezgro može podržavati dodatne IDE kontrolere. Ovi se tada numerišu redom počevši od ide2.

Ako ima SCSI uređaja na sistemu, /proc/scsi/scsi sadrži pregled identifikacionih brojeva uređaja. Na primer, sadržaj bi mogao izgledati ovako:

```
% cat /proc/scsi/scsi
Attached devices:
Host: scsi0 Channel: 00 Id: 00 Lun: 00
  Vendor: QUANTUM Model: ATLAS_V__9_WLS Rev: 0230
  Type: Direct-Access ANSI SCSI revision: 03
Host: scsi0 Channel: 00 Id: 04 Lun: 00
  Vendor: QUANTUM Model: QM39100TD-SW Rev: N491
  Type: Direct-Access ANSI SCSI revision: 02
```

Ovaj računar sadrži jednokanalni SCSI kontroler (obeležen "scsi0"), na kojeg su spojena dva Quantum diskovna uređaja, sa identifikacionim brojevima 0 i 4. Zapis /proc/partitions prikazuje particije prepoznatih diskovnih uređaja. Za svaku particiju, prikaz sadrži njihove *glavne* i *sekundarne* brojeve, broj 1024-bajtnih blokova i ime uređaja koje odgovara toj particiji. Zapis /proc/sys/dev/cdrom/info prikazuje raznovrsne informacije o mogućnostima CDROM uređaja. Polja objašnjavaju sama sebe:

```
% cat /proc/sys/dev/cdrom/info
CD-ROM information, Id: cdrom.c 2.56 1999/09/09

drive name: hdc
drive speed: 48
drive # of slots: 0
Can close tray: 1
Can open tray: 1
Can lock tray: 1
Can change speed: 1
Can select disk: 0
Can read multisession: 1
Can read MCN: 1
Reports media changed: 1
Can play audio: 1
```

7.5.3 Podizanja (mount)

Fajl /proc/mounts pruža pregled podignutih fajl-sistema. Svaka linija odgovara jedinstvenom *maunt-deskriptoru* (eng. *mount - podići)* i prikazuje podignuti uređaj, odredišni direktorijum i druge podatke. Obratite pažnju da /proc/mounts sadrži iste podatke kao i obični fajl /etc/mtab, koji se automatski ažurira komandom mount. Ovo su elementi maunt-deskriptora:

- Prvi element u liniji teksta je podignuti uređaj (pogledajte Poglavlje 6). Kod specijalnih fajl-sistema poput /proc, ovo je none.
- Drugi element je odredišni direktorijum (eng. *mount point - mesto podizanja*), mesto gde se u korenom fajl-sistemu pojavljuje sadržaj ovog uređaja. Za koreni fajl-sistem, odredišni direktorijum je naveden kao /. Za uređaje namenjene virtuelnoj memoriji, odredišni direktorijum je naveden kao swap.

§ Treći element je tip fajl-sistema. Trenutno, većina GNU/Linuks sistema koristi ext2 fajl-sistem za diskovne uređaje, ali DOS-ovi i Windows-ovi uređaji mogu biti podignuti sa drugim tipovima, poput fat ili vfat. Većina CDROM-ova sadrži iso9660 fajl-sistem. Pogledajte man-stranice za komandu mount radi liste tipova fajl-sistema.
§ Četvrti element izlistava zastavice pri podizanju. Ovo su opcije koje su bile naznačene onog trenutka kada se uređaj podizao. Pogledajte man-stranice za komandu mount radi objašnjenja različitih zastavica pri podizanju fajl-sistema raznih tipova. U fajlu /proc/mounts, poslednja dva elementa su uvek 0 i nemaju posebnog značenja. Pogledajte man-stranice za fstab radi detalja o formatu maunt-deskriptora.[6]

GNU/Linuks uključuje funkcije da parsirate maunt-deskriptore; pogledajte man-stranice za funkciju getmntent radi informaciju o korištenju ovih.

7.5.4 Zaključavanja

Sekcija 8.3, "fcntl: Zaključavanja i druge operacije sa fajlovima", opisuje kako koristiti sistemski poziv fcntl radi manipulacije zaključavanja fajlova za čitanje i pisanje. Zapis /proc/locks opisuje sve zaključane fajlove trenutno na sistemu. Svaki red odgovara jednom zaključavanju. Kod katanaca stvorenih funkcijom fcntl, prva dva elementa u redu su POSIX ADVISORY.

Treći je WRITE ili READ, zavisno od tipa zaključavanja. Naredni broj je PID procesa koji drži katanac. Naredna tri broja, razdvojena dvotačkama, su glavni i sekundarni broj uređaja na kojem se nalazi fajl i broj *i-čvora* (*eng. inode number*), koji označava lokaciju fajla u fajl-sistemu. Ostatak linije prikazuje vrednosti interne za jezgro i koje nisu naročito korisne. Pretvaranje podataka iz /proc/locks u nešto korisno zahteva malo istraživanja. Možete videti kako radi /proc/locks, na primer, tako što ćete pokrenuti program iz Listinga 8.2 koji zaključava fajl /tmp/test-file za pisanje.

```
% touch /tmp/test-file
% ./lock-file /tmp/test-file
file /tmp/test-file
opening /tmp/test-file
locking
locked; hit enter to unlock...
```

U drugom prozoru, pogledajte sadržaj fajla /proc/locks.

```
% cat /proc/locks
1: POSIX ADVISORY WRITE 5467 08:05:181288 0 2147483647 d1b5f740 00000000
dfea7d40 00000000 00000000
```

[6] Fajl /etc/fstab izlistava konfiguraciju statičnih podizanja GNU/Linuks sistema

Može biti i dodatnog teksta u rezultatu, koji odgovara katancima stvorenim od strane nekih drugih programa.
U ovom slučaju, 5467 je PID pokrenutog programa lock-file. Koristite ps da vidite šta radi ovaj proces.

```
% ps 5467
PID TTY STAT TIME COMMAND
5467 pts/28 S 0:00 ./lock-file /tmp/test-file
```

Zaključani fajl, /tmp/test-file, je smešten na uređaju sa glavnim i sekundarnim brojevima uređaja 8 i 5, tim redom. Ovi brojevi ovog puta odgovaraju uređaju /dev/sda5.

```
% df /tmp
Filesystem         1k-blocks     Used      Available   Use%   Mounted on
/dev/sda5           8459764      5094292   2935736     63%    /
% ls -l /dev/sda5
brw-rw---- 1 root disk 8, 5 May 5 1998 /dev/sda5
```

Sam fajl /tmp/test-file ima broj i-čvora 181.288 na tom uređaju.

```
% ls --inode /tmp/test-file
181288 /tmp/test-file
```

Pogledajte Sekciju 6.2, “Brojevi uređaja”, radi više informacija o brojevima uređaja.

7.6 Statistike sistema

Dva zapisa u /proc sadrže korisne informacije o statistikama. Fajl /proc/loadavg sadrži informacije o zauzeću sistema. Prva tri broja predstavljaju broj *aktivnih procesa* na sistemu—procesa koji zapravo rade—uprosečeno za poslenjih 1, 5 i 15 minuta. Sledeći zapis pokazuje trenutni broj *procesa koji mogu raditi*—procesa koji su isplanirani da *urade nešto*, za razliku od onih koji bi blokirali u nekom od sistemskih poziva—i ukupni broj procesa na sistemu. Poslednji zapis je PID procesa koji je poslednji nešto radio.
Fajl /proc/uptime sadrži vreme koje je proteklo otkako je sistem upaljen, kao i količinu vremena za koje sistem nije radio ništa. Obe vrednosti su date u decimalnom obliku, u sekundama.

```
% cat /proc/uptime
3248936.18 3072330.49
```

Program u Listingu 7.7 preuzima informacije o ovim vremenima i prikazuje ih u čitljivom formatu.

Listing 7.7 **(*print-uptime.c*) Štampa ukupno vreme rada sistema i ukupno vreme neaktivnosti**

```
#include <stdio.h>

/* Daj pregled količine vremena na standardni izlaz. TIME je količina vremena,
u sekundama, a LABEL je kratak opis. */
void print_time (char* label, long time)
{
```

nastavak na narednoj strani

nastavak sa prethodne strane

```
    /* Konstante za konverziju. */
    const long minute = 60;
    const long hour = minute * 60;
    const long day = hour * 24;
    /* Proizvedi prikaz. */
    printf ("%s: %ld dana, %ld:%02ld:%02ld\n", label, time / day,
    (time % day) / hour, (time % hour) / minute, time % minute);
}

int main ()
{
    FILE* fp;
    double uptime, idle_time;
    /* Procitaj ukupno vreme i ukupno vreme neaktivnosti iz /proc/uptime. */
    fp = fopen ("/proc/uptime", "r");
    fscanf (fp, "%lf %lf\n", &uptime, &idle_time);
    fclose (fp);
    /* Summarize it. */
    print_time ("ukupno vreme ", (long) uptime);
    print_time ("vreme neaktivnosti ", (long) idle_time);
    return 0;
}
```

Komanda uptime i sistemski poziv sysinfo (pogledajte Sekciju 8.14, “sysinfo: Dobavljanje sistemskih statistika”) takođe mogu dobaviti ukupno vreme rada sistema. Komanda uptime takođe prikazuje proseke zauzeća koji se mogu naći u /proc/loadavg.