Prozesuak: kontrola eta kudeaketa

Juanan Pereira
juanan.pereira@ehu.es

Kepa Bengoetxea
kepa.bengoetxea@ehu.es

Mikel Larrea
http://tinyurl.com/yz5wk33

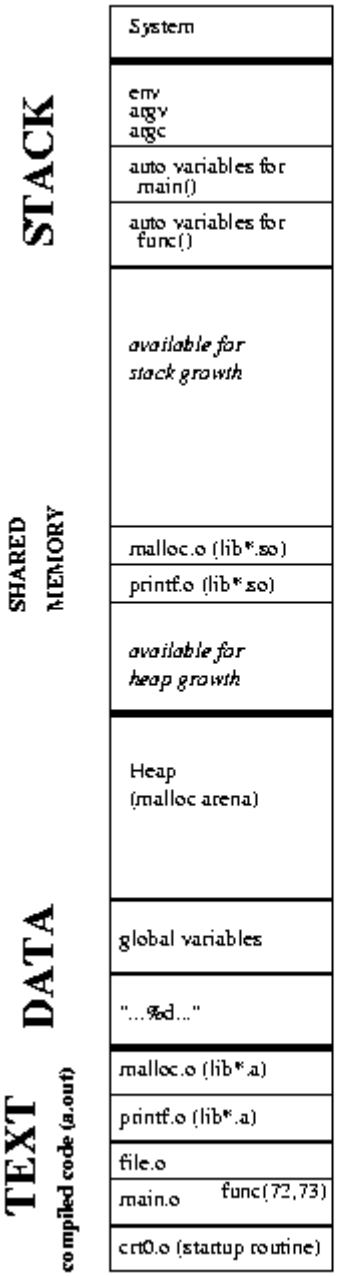
Prozesu kudeaketa

Prozesua: erabiltzaile baten eskaeraren ondorioz martxan dagoen programa. Oinarrizko elementua da sistemarentzat, eta **identifikadore** bat du esleiturik.

Prozesu kudeaketa

A process can broadly be defined into following segments :
•*Stack:* Stack contains all the data that is local to a function like variables, pointers etc. Each function has its own stack. Stack memory is dynamic in the sense that it grows with each function being called.
•*Heap:* Heap segment contains memory that is dynamically requested by the programs for their variables.
•*Data:* All the global and static members become part of this segment.
•*Text:* All the program instructions, hard-coded strings, constant values are a part of this memory area.

Memory Layout (Virtual address space of a C process)



High memory

mfp – frame pointer (for main)

stack pointer (grows downward if func() calls another function)

library functions if dynamically linked (usual case)

brk point

uninitialized data (bss)

initialized data

library functions if statically linked (not usual case)

ra (return address)

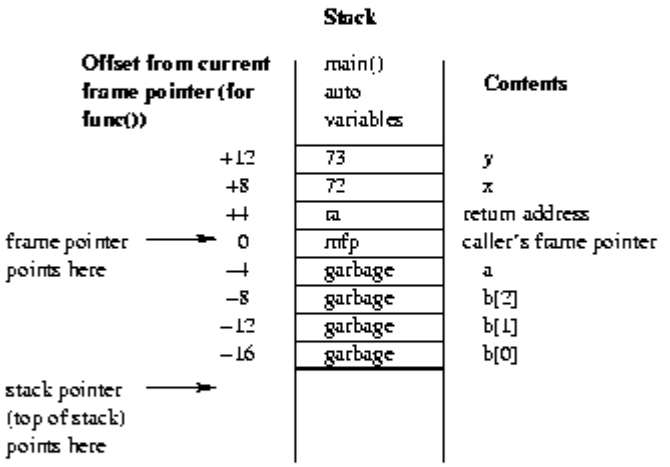
Low memory

Stack illustrated after the call func(72,73) called from main(), assuming func defined by:

```
func(int x, int y) {
  int a;
  int b[3];
 /* no other auto variables */
```

Assumes int = long = char * of size 4 and assumes stack at high address and descending down.

Expanded view of the stack



All auto variables and parameters are referenced via offsets from the frame pointer.

The frame pointer and stack pointer are in registers (for fast access).

When funct returns, the return value is stored in a register. The stack pointer is move to the y location, the code is jumped to the return address (ra), and the frame pointer is set to mfp (the stored value of the caller's frame pointer). The caller moves the return value to the right place.

Prozesu kudeaketa

Prozesuak (egikaritzen ari diren programak) zuhaitz bat osatzen dute.
Zuhaitz horren erroan systemd prozesua (PID 1 duena) kokatzen da.

“pstree” komandoak prozesuen zuhaitza pantailaratuko du.
Aukerak:
- -u prozesua hasi zuenaren izena pantailaratzeko
- -p prozesuaren identifikatzailea bistaratzeko

```
$pstree -p
systemd(1)─┬─ModemManager(871)─┬─{gdbus}(973)
           │                   └─{gmain}(966)
           ├─NetworkManager(909)─┬─dhclient(1390)
           │                     ├─dhclient(2029)
           │                     ├─dnsmasq(1399)
           │                     ├─{gdbus}(1069)
           │                     └─{gmain}(1067)
```

Prozesu kudeaketa

ps komandoa: prozesuen informazioa lortzeko

-u erab01 (erab01 erabiltzailearen prozesuak bistaratu)
-t pts/0 (lehenengo terminaleko prozesuak bistaratu, ze terminalean gauden jakiteko tty komandoa erabili)
-p pid (pid identifikatzailea duen prozesuaren informazioa bistaratu)
-a beste erabiltzaileen prozesuak bistaratu (eta ez soilik uneko erabiltzailearenak)
-x sistemaren prozesuak ere bistaratu
-f full (prozesuen zuhaitza pantailaratu)

Prozesu kudeaketa

ps -aux komandoak hurrengo informazioa pantailaratu dezake:

```
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
kepa      3496  0.0  0.0 210380  7820 ?        Sl   08:53   0:00 /usr/lib/libreoffice/program/oosplash --impress
file:///media/datos/Dropbox/docencia/isobilbo/ISO17_18/Enlace%20hacia
%20Gaiak/5.ProzesuenKudeaketa/teoria/5_2_prozesuak_laburtua.odp
kepa      3515  0.5  0.9 1427988 152876 ?      Sl   08:53   0:05 /usr/lib/libreoffice/program/soffice.bin --impress
file:///media/datos/Dropbox/docencia/isobilbo/ISO17_18/Enlace%20hacia
%20Gaiak/5.ProzesuenKudeaketa/teoria/5_2_prozesuak_laburtua.odp --splash-pipe=5
kepa      3570  0.3  0.2 666136 38504 ?        Sl   09:02   0:01 /usr/lib/gnome-terminal/gnome-terminal-server
kepa      3577  0.0  0.0  25296  5372 pts/2    Ss   09:02   0:00 bash
kepa      3690  0.0  0.0  39936  3340 pts/2    R+   09:11   0:00 ps -aux
kepa      3691  0.0  0.0  16764   940 pts/2    S+   09:11   0:00 grep --color=auto kepa
```

*RSS the Resident Set Size and is used to show how much memory is allocated to that process and is in RAM. It does not include memory that is swapped out. It does include memory from shared libraries as long as the pages from those libraries are actually in memory. It does include all stack and heap memory.
*VSZ is the Virtual Memory Size. It includes all memory that the process can access, including memory that is swapped out and memory that is from shared libraries.
For example: if process A has a 500K binary and is linked to 2500K of shared libraries, has 200K of stack/heap allocations of which 100K is actually in memory (rest is swapped), and has only actually loaded 1000K of the shared libraries and 400K of its own binary then:
RSS: 400K + 1000K + 100K = 1500K
VSZ: 500K + 2500K + 200K = 3200K

Prozesu kudeaketa

Seinaleak: kernelak prozesuei bidaltzen dizkien mezuak. Seinaleek osoko identifikatzaile bat dute. **kill**: prozesu bati seinale bat bidaltzeko komandoa. Adibidez: **kill -9 PID**

kill -l

1) SIGHUP 2) SIGINT 3) SIGQUIT 4) SIGILL 5) SIGTRAP

6) SIGABRT 7) SIGBUS 8) SIGFPE 9) SIGKILL 10) SIGUSR1

11) SIGSEGV 12) SIGUSR2 13) SIGPIPE 14) SIGALRM 15) SIGTERM

16) SIGSTKFLT 17) SIGCHLD 18) SIGCONT 19) SIGSTOP 20) SIGTSTP

21) SIGTTIN 22) SIGTTOU

Prozesu kudeaketa

- Erabiltzaile arrunt batentzat benetan interesgarriak diren seinaleak honakoak dira:

-KILL: PID prozesua amaitu (baldintzarik gabe eta berehala)

-HUP: PID prozesuari bere konfigurazio fitxategia irakurtarazi

-TERM: PID prozesua amaitu (-TERM-ek prozesuari ondo bukatzeko aukera ematen dio. -KILL-ek, berriz, ez)

-STOP: =Ctrl+Z (prozesua eten, lotan utzi)

-CONT: PID prozesuari jarraitzeko seinalea bidali (orokorrean, -STOP seinalea jaso eta gero lantzen ohi den seinalea)

Ikus:http://en.wikipedia.org/wiki/SIGHUP

Prozesu kudeaketa

- Adibideak

Kill komandoa balio lehenetsi bezala TERM seinalea bidaliko du.

- $ kill 4541
- $ kill -15 4541
- $ kill -TERM 4541
- $ kill -s SIGTERM 4541

- $ kill -9 3454
- $ kill -KILL 3454
- $ kill -s SIGKILL 3454

Ikus:http://en.wikipedia.org/wiki/SIGHUP

Prozesu kudeaketa

top: prozesuak baliabideen kontsumoaren arabera ikusteko (segundo batzuen maiztasun finkoarekin eguneratzen da informazioa)

Laguntza: “h” edo “?”
Ordenatzeko irizpidea hautatu : ">" edo "<" sakatu
Txikietatik handienera edo alderantziz ordenatu: R
Nahi ditugun zutabeak hautatu: "f"
Koloreak aldatu (oso gomendagarria): z
Azpimarratu ordenatzeko irizpidea: x

Prozesu kudeaketa

nohup komandoa:

Seme prozesu bat hil baino lehen aita hiltzen bada, semea systemd prozesuak adoptatuko du. Sabuespen bakarra “bash” aitarekin bidalitako prozesu semeak. “Bash” aita itxiz gero, bere “bash”etik bidalitako prozesu seme guztiak hilko ditu. Hori ekiditzeko, nohup komandoa erabil daiteke.

nohup komandoa

Noiz erabili? Zerbitzari batean komando edo ataza bat bidaltzerakoan

$ ssh erabiltzailea@helbidea

nohup komandoa

Prozesu kudeaketa

nice komandoa: prozesu baten lehentasuna aldatu.

Sintaxia: (egikaritu behar den komando baten lehentasuna aldatu)

nice -n <x> komandoa

<x> -20 eta +19 bitartean egongo da, -20 lehentasun handiena izanik (cpu denbora gehien hartuko duena)

root erabiltzaileak esleitu ditzake 0 baino txikiagoak diren lehentasunak.

Jada egikaritu den komando baten lehentasuna aldatu nahi bada:

renice <x> PID

Prozesu kudeaketa

nice komandoa. Adibidea:

```
$yes > /dev/null &
[1] 9862
$ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S   500  9841  5770  0  75   0 -  1409 wait   pts/1    00:00:00 bash
0 R   500  9862  9841 65  85   0 -  1203 -      pts/1    00:00:03 yes
0 R   500  9863  9841  0  76   0 -   597 -      pts/1    00:00:00 ps
$ renice +19 9862
 9862: prioridad antigua 0, nueva prioridad 19
$ sudo  renice -20 9862 (como root, para menores de 0)
9862: prioridad antigua 19, nueva prioridad -20
kill -9 9862
```

Prozesu kudeaketa

```
bcplemza@B900112:~$ ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1000  2466  2461  7  80   0 -  6938 wait   pts/1    00:00:00 bash
0 R  1000  2521  2466  0  80   0 -  3379 -      pts/1    00:00:00 ps
bcplemza@B900112:~$ nice -n 5 xclock &
[1] 2522
bcplemza@B900112:~$ ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1000  2466  2461  1  80   0 -  6938 wait   pts/1    00:00:00 bash
0 S  1000  2522  2466  0  85   5 - 16049 poll_s pts/1    00:00:00 xclock
0 R  1000  2523  2466  0  80   0 -  3379 -      pts/1    00:00:00 ps
bcplemza@B900112:~$ renice 12 2522
2522: prioridad antigua 5, nueva prioridad 12
bcplemza@B900112:~$ ps -l
F S   UID   PID  PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1000  2466  2461  0  80   0 -  6938 wait   pts/1    00:00:00 bash
0 S  1000  2522  2466  0  92  12 - 16049 poll_s pts/1    00:00:00 xclock
0 R  1000  2525  2466  0  80   0 -  3379 -      pts/1    00:00:00 ps
bcplemza@B900112:~$
```

Prozesu kudeaketa

tee komandoa: sarrera estandarretik irakurritakoa irteera estandarrean eta argumentu bezala emandako fitxategian kopiatzen du

Adibidea:

```
kepa@cox:/tmp$   who | tee konektatuta | sort | tee ordenatua
kepa   pts/0        2009-11-03 21:30 (:0.0)
kepa   pts/1        2009-11-03 22:01 (:0.0)
kepa   tty7         2009-11-03 20:31 (:0)
```

Prozesu kontrola

sleep komandoa: eten bat sortzen du, guk parametroan ezarritako denbora tarte bitartean

Adibidea:

```
sleep 15m 10s ; mplayer iratzargailua.mp3
```

15 minutu eta 10 segundu barru, iratzargailua.mp3 soinu fitxategia jo.

Prozesu kontrola

at komandoa: prozesuen denbora jakin batean behin exekutatzeko erabiltzen da (atd deabrua haietaz arduratzen da).
Instalatzeko: sudo aptitude install at

Sintaxia:

at <ordua> <agindua>

Aukerak:

at -l (programatuta dauden atazen zerrenda eskatu)

at -d <n> (n zenbakia duen ataza zerrendatik kendu)

man at (-f aukera ikusteko)

Prozesu kontrola

Adibidea:

```
$ at now +2 minutes -f eskripta.sh
$ at now +2 minutes

at> echo kaixo > /home/kepa/Desktop/kaixo.txt
at> <EOT>   <--- Ctrl+D

job 1 at Sun Nov 15 19:50:00 2009
```

Prozesu kontrola

Atd zerbitzua kudeatu:

```
sudo service atd stop          #gelditu
sudo service atd start         #hasieratu
sudo service atd status  #egoera
```

Prozesu kontrola

crontab komandoa: prozesuak aldizka exekutatzeko erabiltzen da (crond deabrua haietaz arduratzen da).

·Zertarako erabiltzen da?

- Katalogo batzuen segurtasun kopiak egiteko.
- Minutu gutxi batzuetara erabiltzaile konektatuen informazioa bildu egiteko.
- Sistemaren eguneraketa egiteko
- ...

Prozesu kontrola

- Sintaxia: **crontab -e** , “/var/spool/cron/crontabs/kepa” katalogoan gordeko du, fitxategi honen egitura, hilara bakoitzean 6 zutabe hutsune bategaz banatuta:

  1.-Minutuak 0-59

  2.-Orduak 0-23

  3.-Hileko zein eguna 1-31

  4.-Hila 1-12

  5.-Asteko eguna 0 (igandea) eta 6 (larunbata)

  6.-komandoa edo ataza

Prozesu kontrola

- Adibidez:crontab -e

```
# m h  dom mon dow   command
30 9 * * * touch /home/kepa/Escritorio/proba.txt
```

Prozesu kontrola

- "crontab"-aren aukerak:
  - -l ikusi nire crontab lerroak
  - -e editatu edo borratu nire crontab lerroak
  - -r nire crontab fitxategia ezabatu
  - -u <erabiltzailea> (root edo sudo bezala soilik)

Prozesu kontrola

- Hartu dezakeen baloreak:

  * edozein

  2-6(2tik 6ra)

  2,4,6(soilik 2,4 eta 6)

  */5 (5 minuturo, 5 orduro...)

Prozesu kontrola

·Gure lana errazteko definitutako katalogo batzuk daude /etc barruan. Katalogo horretan jartzen dituzun exekutagarriak (scriptak, komandoak eta abar …) orduro, egunero, eta abar exekutatuko dira. Katalogoak, cron.hourly, cron.daily.... dira.

```
 $less /etc/crontab
SHELL=/bin/bash
PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin
# m h dom mon dow user  command
17 *    * * *   root    cd / && run-parts --report /etc/cron.hourly
25 6    * * *   root    test -x /usr/sbin/anacron || ( cd / && run-parts --report /etc/cron.daily )
```

Oharra: anacron martxan egonez gero, /etc/cron.daily katalogoan dauden scriptak edo exekutagarriak ez dira exekutatuko.

Prozesu kontrola

- Adibidez: “proba.sh” scripta egunero exekutatzeko
  cd /etc/cron.daily/

  gedit proba.sh

  echo kaixo > /home/kepa/Escritorio/kaixo.txt

- Loga edo bitakora: exekutatu den jakiteko. less /var/log/syslog | grep cron.hourly

Nov 30 10:17:01 kepa-laptop CRON[9303]: (root) CMD (cd / && run-parts --report /etc/cron.hourly)

Nov 30 11:17:01 kepa-laptop CRON[9824]: (root) CMD (cd / && run-parts --report /etc/cron.hourly)

...

Crontab & vi

sudo apt-get remove vim.tiny

sudo apt-get install vim

Command mode: The ESC key can end a command

Insert mode: Text is inserted. The ESC key ends insert mode and returns you to command mode. One can enter insert mode with the "i" (insert), "a" (insert after), "A" (insert at end of line), "o" (open new line after current line) or "O" (Open line above current line) commands.

Command line mode: One enters this mode by typing ":" which puts the command line:”:**q!”**(Ignore changes and quit. No changes from last write will be saved.)**”:wq”** (Save (write) changes to current file and quit.

Crontab & gedit

Añadir la siguiente línea en el archivo .bashrc

```
export EDITOR=gedit;
source .bashrc
env
crontab -e
# m h  dom mon dow   command
38 9 * * * touch /home/euiti/Escritorio/prueba.txt
```

Prozesu kontrola

- "**anacron**" komandoa: crontab bezalakoa da, baina ordenagailua momentu oro martxan ez dauden kasuetarako erabiltzen da.

Konfigurazio fitxategia: ***/etc/anacrontab:***

SHELL=/bin/sh

PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin

MAILTO=root

#aldia atzerapena ataza-identificador-tarea komandoa

1 5 cron.daily nice run-parts --report /etc/cron.daily

7 10 cron.weekly nice run-parts --report /etc/cron.weekly

@monthly 15 cron.monthly nice run-parts --report /etc/cron.monthly

Prozesu kontrola

- Anacron komandoa exekutatzerakoan, **aldiro** zutabean dagoan azkenengo **n** egunetan komandoa exekutatua ez bada, **atzerapen minutuak** itxaron ostean ataza hori exekutatuko du.

Logak: less /var/log/syslog | grep anacron

Nov 30 10:07:13 kepa-laptop anacron[1061]: Job `cron.daily' terminated

Nov 30 10:07:13 kepa-laptop anacron[1061]: Normal exit (1 job run)

Anacron ordenagailua abiatzerakoan exekutatzen da eta gero amaitzen da.

Jakiteko noiz exekutatu den azkenego aldiz: ls /var/spool/anacron/

cron.daily cron.monthly cron.weekly

sudo less /var/spool/anacron/cron.daily --> 20170322

Prozesu kontrola

Gehiago jakiteko:

https://help.ubuntu.com/community/CronHowto