SO Curs 3

Cuprins

Suport de curs

OSCE
- Capitolul 3 – Process

MOS
- Capitolul 2 – Processes and Threads
- Secțiunea 1

LSP
- Capitolul 5 – Process Management
- Capitolul 6 – Advanced Process Management
- Capitolul 9 – Signals

WSP
- Capitolul 6 – Process Management
- Capitolul 11 – Interprocess Communication

Cuvinte cheie

proces
program
ierarhia de procese
PCB
atribute de proces
starea unui proces
descriptori de resurse
multitasking
context switch
planificator
crearea unui proces
terminarea unui proces
asteptarea unui proces
fork
exec
CreateProcess
IPC
semnale
pipe-uri

API de procese

shell
ANSI C
POSIX
Win32
Python
Java

Atributele unui proces

Process Control Block (PCB)
Resursele unui proces
Ierarhia de procese
Starea unui proces

Planificarea proceselor

Paralelism la execuție
Tranziții între stări
Schimbare de context
Planificatorul de procese

Comunicarea interproces

Semnale
Pipe-uri
Memorie partajata
Message passing
Socketi

Procese

Ce este un proces?
Programe si procese
De ce procese?
Spațiul de adrese al unui proces
Tipuri de procese

Operatii cu procese

Crearea unui proces
Încheierea unui proces
Terminarea unui proces
Așteptarea unui proces

Lucrul cu procese în ANSI C

Lucrul cu procese în POSIX

fork si exec

Lucrul cu procese în POSIX (2)

Lucrul cu procese în Win32

Lucrul cu procese în Python

subprocess module

Lucrul cu procese în Java

ProcessBuilder

Lucrul cu procese în shell

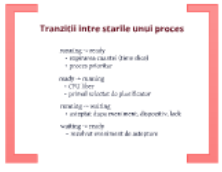
Descriptorii de resurse

Resursele unui proces

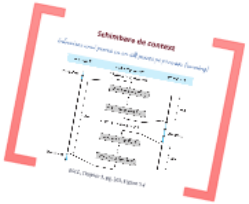
Ierarhia de procese

Multitasking

Tranziții între starile unui proces

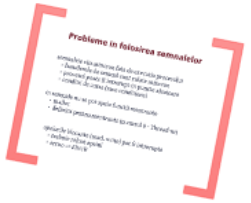


Schimbare de context



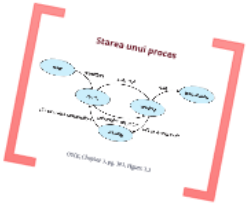
Semnale

Probleme în folosirea semnalelor



Process Control Block

Starea unui proces



Paralelism la nivel de SO

Planificarea proceselor

De ce procese?

Program vs. Proces

Spațiul de adrese al unui proces

Încheierea unui proces

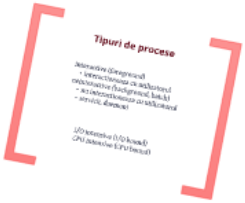
fork si exec în Unix

Terminarea unui proces

Așteptarea unui proces

Ce este un proces?

Tipuri de procese



Crearea unui proces

Pipe-uri

Memorie partajata

Message Passing

Socketi

SO Curs 3

Procese

Cuprins

Suport de curs

OSCE
- Capitolul 3 – Process

MOS
- Capitolul 2 – Processes and Threads
- Secțiunea 1

LSP
- Capitolul 5 – Process Management
- Capitolul 6 – Advanced Process Management
- Capitolul 9 – Signals

WSP
- Capitolul 6 – Process Management
- Capitolul 11 – Interprocess Communication

Cuvinte cheie

proces
program
ierarhia de procese
PCB
atribute de proces
starea unui proces
descriptori de resurse
multitasking
context switch
planificator
crearea unui proces
terminarea unui proces
asteptarea unui proces
fork
exec
CreateProcess
IPC
semnale
pipe-uri

API de procese

shell
ANSI C
POSIX
Win32
Python
Java

Lucrul cu procese în ANSI C

Lucrul cu procese în POSIX

fork si exec

Lucrul cu procese în POSIX (2)

Lucrul cu procese în Win32

Lucrul cu procese în Python

subprocess module

Lucrul cu procese în Java

ProcessBuilder

Lucrul cu procese în shell

Descriptorii de resurse

Ierarhia de procese

Resursele unui proces

Tranziții între stările unui proces

Multitasking

Schimbare de context

Semnale

Probleme în folosirea semnalelor

Atributele unui proces

Process Control Block (PCB)
Resursele unui proces
Ierarhia de procese
Starea unui proces

Starea unui proces

Process Control Block

Paralelism la nivel de SO

Planificarea proceselor

Paralelism la execuție
Tranziții între stări
Schimbare de context
Planificatorul de procese

Planificarea proceselor

Comunicarea interproces

IPC (Inter-Process Communication)

Semnale
Pipe-uri
Memorie partajata
Message passing
Socketi

De ce procese?

Program vs. Proces

Spațiul de adrese al unui proces

Încheierea unui proces

Terminarea unui proces

Pipe-uri

Procese

Ce este un proces?
Programe si procese
De ce procese?
Spațiul de adrese al unui proces
Tipuri de procese

Ce este un proces?

Tipuri de procese

fork si exec in Unix

Crearea unui proces

Operatii cu procese

Crearea unui proces
Încheierea unui proces
Terminarea unui proces
Așteptarea unui proces

Așteptarea unui proces

Procese zombie și orfane

Memorie partajata

Message Passing

Socketi

Suport de curs

OSCE
- Capitolul 3 – Process

MOS
- Capitolul 2 – Processes and Threads
  - Sectiunea 1

LSP
- Capitolul 5 – Process Management
- Capitolul 6 - Advanced Process Management
- Capitolul 9 – Signals

WSP
- Capitolul 6 – Process Management
- Capitolul 11 – Interprocess Communication

Cuprins

Procese

Ce este un proces?
Programe si procese
De ce procese?
Spatiul de adrese al unui proces
Tipuri de procese

Ce este un proces?

Un program aflat in executie
Instanta a unui program

Abstractizarea actiunii in sistemul de operare

Abstractizare peste procesor

Ce se face si cu ce resurse

Program vs. Proces

pasiv
executabil
date, cod
cale in sistemul de fisiere
imaginea unui proces

activ
memorie & CPU
date, cod, heap, stiva
PID (Process ID)
instanta unui program
resurse
spatiu de adresa

Care este asocierea program - proces?

De ce procese?

Mod de abstractizare a actiunii

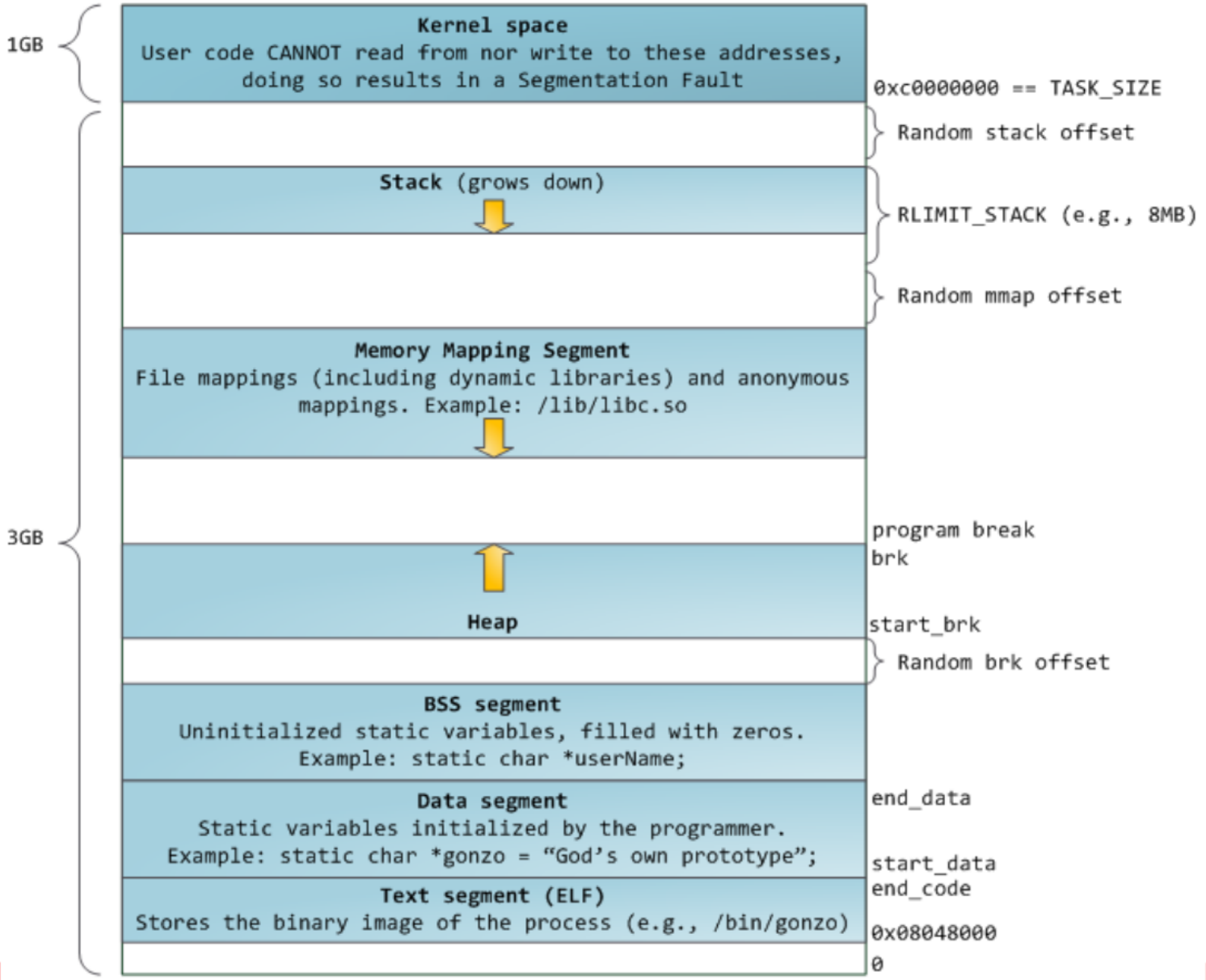
Unitatea de lucru in sistemul de operare

Agregarea resurselor folosite

Abstractizare peste CPU si memorie

Spatiul de adrese al unui proces

vezi si pmap



http://duartes.org/gustavo/blog/post/anatomy-of-a-program-in-memory

Tipuri de procese

interactive (foreground)
- interactioneaza cu utilizatorul

neinteractive (background, batch)
- nu interactioneaza cu utilizatorul
- servicii, daemoni

I/O intensive (I/O bound)
CPU intensive (CPU bound)

Atributele unui proces

Process Control Block (PCB)

Resursele unui proces

Ierarhia de procese

Starea unui proces

Process Control Block

PCB

Structura ce descrie procesul si atributele acestuia

- struct task_struct in nucleul Linux
- EPROCESS/KPROCESS in nucleul Windows

PID
spatiul de adrese (zonele de memorie)
tabela de descriptori de fisier
masca de semnale
informatii de securitate (user id, group id, capabilitati)
referinte la alte procese
informatii de monitorizare si stare

Resursele unui proces

resursele fizice abstractizate in resurse logice

- disc - fisiere
- retea - socketi

operatii cu resurse: deschidere, inchidere, citire, scriere

resursele au un identificator (descriptor) si o stare

private sau partajate cu alte procese

Descriptorii de resurse

file descriptori, handle-uri, tabele

unici la nivelul unui proces (namespace)

- descriptorul 4 din procesul P1 refera altceva in procesul P2
- poate fi valid intr-un proces dar nevalid in altul

Ierarhia de procese

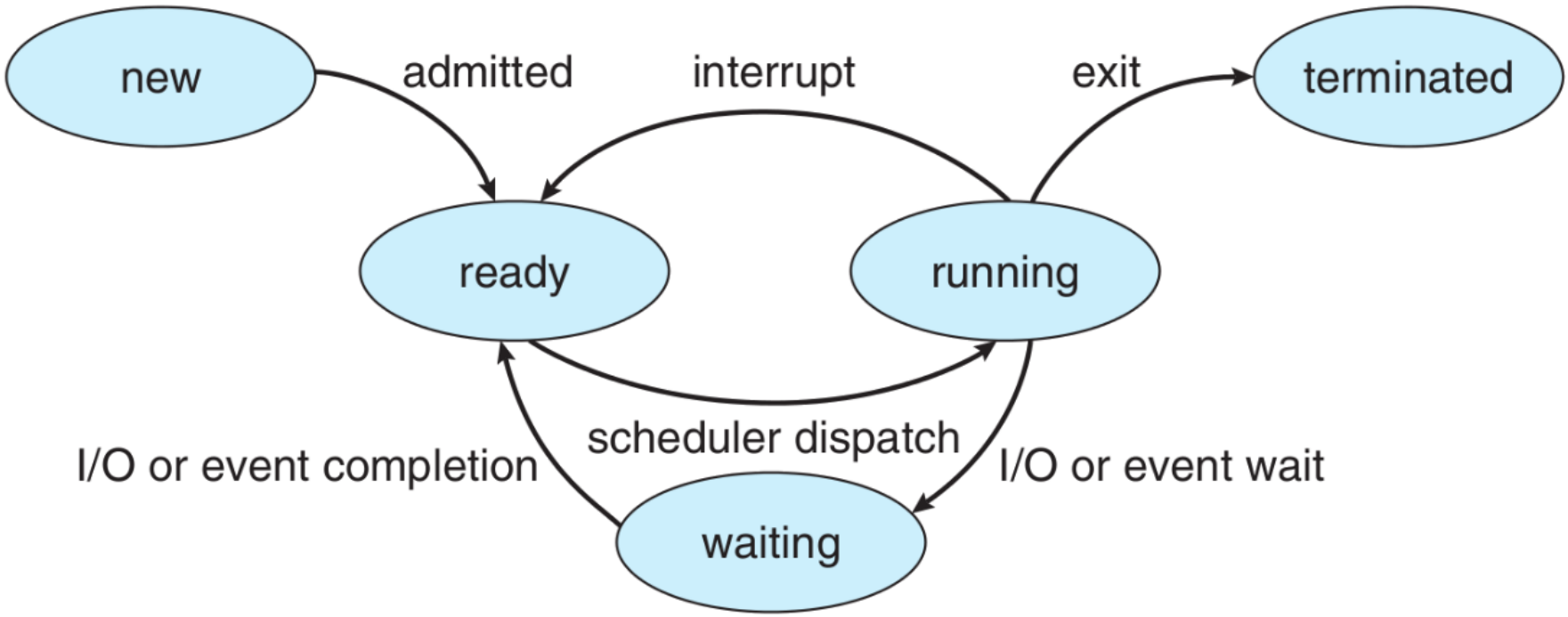
un proces are un proces parinte
un proces are mai multe procese copil

in Unix, init este la radacina ierarhiei de procese

un proces are un PID si un PPID (parent process ID)

pstree, ps -H, Process Explorer

Starea unui proces



OSCE, Chapter 3, pg. 101, Figure 3.2

Planificarea proceselor

Paralelism la executie
Tranzitii intre stari
Schimbare de context
Planificatorul de procese

Paralelism la nivel de SO

mai multe procese la nivelul SO
SO mediaza accesul proceselor la procesor

cu adevarat simultan ruleaza N procese

- N = numar de procesoare

pseudoparalelism: procesele sunt schimbate rapid

paralelism efectiv: sisteme multiprocesor, multicore

Multitasking

procesele sunt schimbate rapid de SO
- cuanta de timp de rulare (time slice)
- la expirarea cuantei, procesul este inlocuit de pe procesor

permite prezenta simultana a mai multor utilizatori

nu este garantata executia completa a unei secvente
- un proces va fi intrerupt la expirarea cuantei

expirarea cuantei declanseaza inlocuirea (schimbare de context)

Tranzitii intre starile unui proces

running -> ready
- expirarea cuantei (time slice)
- proces prioritar

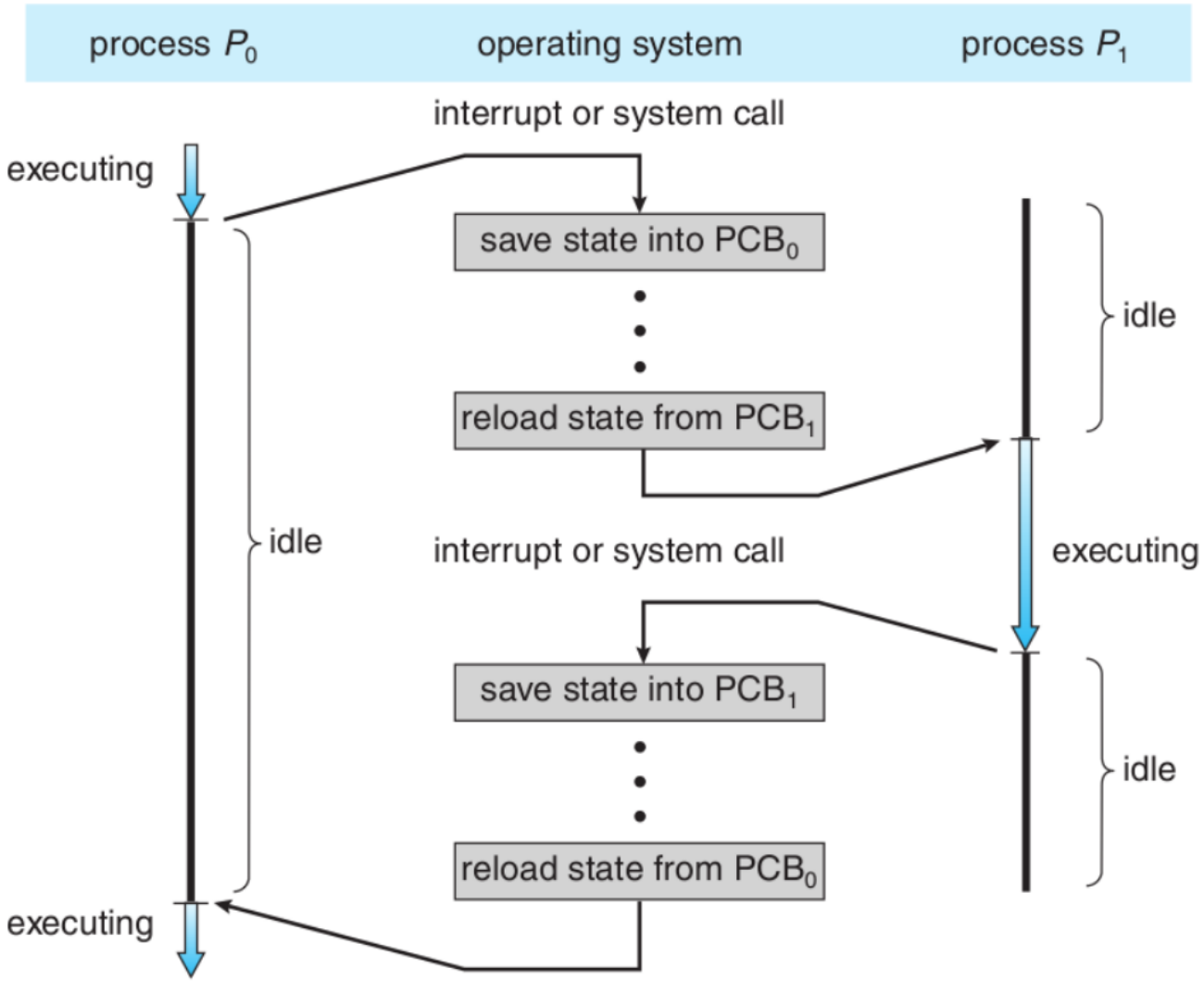
ready -> running
- CPU liber
- primul selectat de planificator

running -> waiting
- asteptat dupa eveniment, dispozitiv, lock

waiting -> ready
- rezolvat eveniment de asteptare

Schimbare de context

înlocuirea unui proces cu un alt proces pe procesor (running)



OSCE, Chapter 3, pg. 103, Figure 3.4

Planificarea proceselor

scheduling, scheduler

subsistem al nucleului SO (process scheduling, process management)

planificarea proceselor pe procesoare
- alegerea celui mai potrivit proces din coada ready
- declansat de o schimbare de context

cele mai potrivite procese
- prioritati
- euristici specifice
- algoritmi de planificare

criterii de planificare
- eficienta, performanta (throughput)
- echitatate (fairness)

mai multe in cursul 4 – Planificarea executiei

Operatii cu procese

Crearea unui proces
Incheierea unui proces
Terminarea unui proces
Asteptarea unui proces

Crearea unui proces

dintr-un proces existent
un proces existent (parinte) creeaza un nou proces (copil)

procesul nou are la baza un program/executabil
executabilului i se asociaza resurse dinamice si devine proces
creat cu ajutorul loader-ului (ld-linux.so pe Linux)

input: executabil si argumente
output: un nou proces, resurse noi, un nou PID

Cine este procesul parinte la rularea comenzii ls?

fork si exec in Unix

fork creeaza un proces identic cu primul (o clona)

- procesul parinte si copil sunt identice
- in general au multe resurse partajate
- input: nimic
- output: un nou proces

exec "transforma" procesul copil pornind de la un executabil

- resursele sunt acum unice fiecarui proces
- procesul "transformat" are un alt spatiu de adresa
- acum se invoca loader-ul
- PID-ul procesului nu se schimba
- input: executabil cu argumente
- output: actualizarea resurselor procesului

Incheierea unui proces

procesul isi incheie executia

- decizie a procesului

  - ajunge la finalul programului (finele lui main)
  - se apeleaza exit()

input: valoare de retur
output: proces incheiat, nu mai exista in sistem

Terminarea unui proces

procesul este terminat de alta entitate

- nucleul SO
- un alt proces

  - primirea unui semnal (Unix)
  - primirea unei exceptii (Windows)

CTRL+C, End Task, Segmentation Fault, Division by Zero etc.

- transmiterea unui semnal sau a unei exceptii

input: PID, cauza
output: proces inexistent in sistem

Asteptarea unui proces

procesele isi sincronizeaza executia
- un proces asteapta incheierea altui proces
- "waiting for another process"

dupa incheierea unui proces, exista informatii remanente
- modul in care s-a incheiat procesul (valoarea de retur)

functiile de asteptare ofera informatiile remanente
input: PID-ul procesului asteptat
output: informatii remanente dupa incheierea procesului

Procese zombie si orfane

un proces copil este asteptat de un proces parinte

probleme:
- parintele moare
- copilul moare inainte sa il astepte parintele

proces zombie
- proces care a murit inainte sa fie asteptat
- ramane in starea zombie pana la asteptare (wait)
- este nevoie de informatie reziduala (valoarea de retur)

proces orfan
- proces caruia i-a murit parintele
- procesul este adoptat de init

API de procese

shell
ANSI C
POSIX
Win32
Python
Java

Lucrul cu procese in shell

creare: ls, mv, top, ifconfig

incheiere: dupa incheierea comenzii

asteptare: implicit, shell-ul se blocheaza
- operatorul & (run in background)

terminare: kill sau CTRL+C, CTRL+Z, CTRL+\

Care este relatia de rudenie intre shell si procesele create?
Ce face comanda "exec ls"?

Lucrul cu procese in ANSI C

```
system("ps -u student");
```

```
FILE *f = popen("ps -u student", "r");
pclose(f);
```

- nu e ANSI, e POSIX, dar are suport si pe Windows

Lucrul cu procese in POSIX

fork si exec

```
pid = fork();
switch (pid) {
        case -1: /* fork failed */
                perror("fork");
                exit(EXIT_FAILURE);
        case 0: /* procesul copil */
                execlp("/usr/bin/ps", "ps", "-u", "student", NULL);
        default: /* procesul printe */
                printf("Created process with pid %d\n", pid);
}
```

fork e apelat o data si se intoarce de doua ori

- intoarce 0 in copil
- PID-ul procesului copil in parinte
- Cum afla procesul copil PID-ul parintelui?
- Ce face procesul parinte cu PID-ul copilului?

Lucrul cu procese in POSIX (2)

```
exit(EXIT_SUCCESS);


kill($PID, SIGTERM);


pid = wait(&status);
pid = waitpid(-1, &status, 0); /* echivalent */
```

Lucrul cu procese in Win32

PROCESS_INFORMATION pi;
CreateProcess(NULL, “notepad”, NULL, NULL, ..., &pi);

- nu exista fork si exec (separat) pe Windows

exit(0);

TerminateProcess(pi.hProcess, 123);

WaitForSingleObject(pi.hProcess, INFINITE);
GetExitCodeProcess(pi.hProcess, &retValue);

Lucrul cu procese in Python

subprocess module

```
subprocess.call(["ps", "-u", "student"])
```

```
p = subprocess.Popen(["ps", "-u", "student"],
        shell=True, bufsize=bufsize,
        stdin=PIPE, stdout=PIPE, stderr=PIPE, close_fds=True)
(child_stdin,
 child_stdout,
 child_stderr) = (p.stdin, p.stdout, p.stderr)
```

Lucrul cu procese in Java

ProcessBuilder

```
ProcessBuilder builder = new ProcessBuilder("ps", "-u", "student");

Process p = builder.start();
InputStream is = p.getInputStream();
OutputStream os = p.getOutputStream();
```

echivalent cu subprocess (Python) si popen()

Comunicarea interproces

IPC (Inter-Process Communication)

Semnale
Pipe-uri
Memorie partajata
Message passing
Socketi

Semnale

doar pe Unix; pe Windows - exceptii

notificarea unei situatii neprevazute

un semnal este transmis de un proces sau de kernel altui proces

identificat printr-un numar
kill -l

handler de tratare a semnalului

- de multe ori, in mod implicit, handler-ul este "omoara procesul"

Probleme in folosirea semnalelor

semnalele vin asincron fata de executia procesului
- handlerele de semnal sunt rulate asincron
- procesul poate fi intrerupt in puncte aleatoare
- conditii de cursa (race conditions)

in semnale nu se pot apela functii reentrante
- malloc
- definitii pentru reentrante in cursul 8 - Thread-uri

apelurile blocante (read, write) pot fi intrerupte
- trebuie reluat apelul
- errno == EINTR

Pipe-uri

comunicare simpla intre procese inrudite
pipe = buffer la nivelul nucleului

operatii de open, read, write, close

un pipe = 2 file descriptori (unul de citire, unul de scriere)
- un proces scrie un proces citeste

implementarea operatorului | (pipe)

pipe-uri cu nume (FIFO)
- intrare in sistemul de fisiere
- comunicare intre procese neinrudite

Memorie partajata

bucati din spatiul de adrese al mai multor procese refera acelasi spatiu de memorie fizica (RAM)

accesul se face pe bază de pointeri (adrese)

- nu e nevoie de apeluri de sistem pentru acces

avantaj: rapida
dezavantaj: necesita sincronizare

Message Passing

formatarea datelor ca mesaje cu sursa si destinatie
multe biblioteci (in diverse limbaje) pentru message passing

util in medii distribuite

livrare directa procesului
livrare indirecta intr-un mailslot

operatii de tipul send/receive
- sincrone sau asincrone

Socketi

comunicare in retea (BSD sockets)
comunicare locala (Unix sockets)

socket API: socket, connect, listen, send, recv

comunicare de tip "teava": byte stream

Cuvinte cheie

proces
program
ierarhia de procese
PCB
atribute de proces
starea unui proces
descriptori de resurse
multitasking
context switch
planificator

crearea unui proces
terminarea unui proces
asteptarea unui proces
fork
exec
CreateProcess
IPC
semnale
pipe-uri