Procese

SO: Curs 3

Sisteme de fișiere

- De ce este nevoie de sisteme de fișiere?
- Ce este un fișier? Perspective
- Cu ce diferă un fișier obișnuit de un director?
- Ce este un descriptor de fișier?
- Ce poate fi referit printr-un descriptor de fișier?
- Ce informații conține o structură de fișier deschis?
- Ce operații (API C) pe fișier modifică cursorul de fișier?
- Ce operații (API C) pe fișier modifică câmpul size din metadatele fișierului?
- Ce se întâmplă cu tabela de descriptori la redirectare?

Ce este un proces?

- Modul în care se execută acțiuni în sistemul de operare
- Un program aflat în execuție
- Încapsularea/abstractizarea execuției în SO
- Abstractizare peste procesor

De ce procese?

- Ca să putem efectua acțiuni în SO
- Ca să avem o separare a acțiunilor: un proces face ceva, altul face altceva
- Ca să avem izolare: un proces nu încurcă alt proces
- Pentru modularizare: când ceva pică sau nu merge, să putem determina rapid problema

Cuprins

- Rolul proceselor
- Atributele unui proces
- Planificarea proceselor
- Crearea unui proces
- Alte operații cu procese
- API pentru lucrul cu procese

Suport de curs

- OSCE
  - Capitolul 3: Processes
- MOS
  - Capitolul 2: Processes and Threads (Secțiunea 1)
- LSP
  - Capitolul 5: Process Management
  - Capitolul 6: Advanced Process Management
- WSP
  - Capitolul 6: Process Management

ROLUL PROCESELOR

Ce face un proces?

- Rulează instrucțiuni pe procesor
  - Instrucțiunile sunt în memoria procesului (cod)
- Lucrează cu informații de la input/output
  - Fișiere, rețea, tastatură, monitor
- Interacționează cu alte procese
  - Comunicare, partajare resurse

Ce conține un proces?

- Memorie: cod, date
- Tabelă de descriptori de fișiere
- Legături către alte procese (proces părinte, procese copil)

Proces și procesor

- Un proces rulează instrucțiuni pe procesor
- Procesele sistemului au nevoie de procesoare
- De obicei procesele sistemului sunt mai multe decât procesoarele sistemului
  - E nevoie de planificarea proceselor pe procesoare

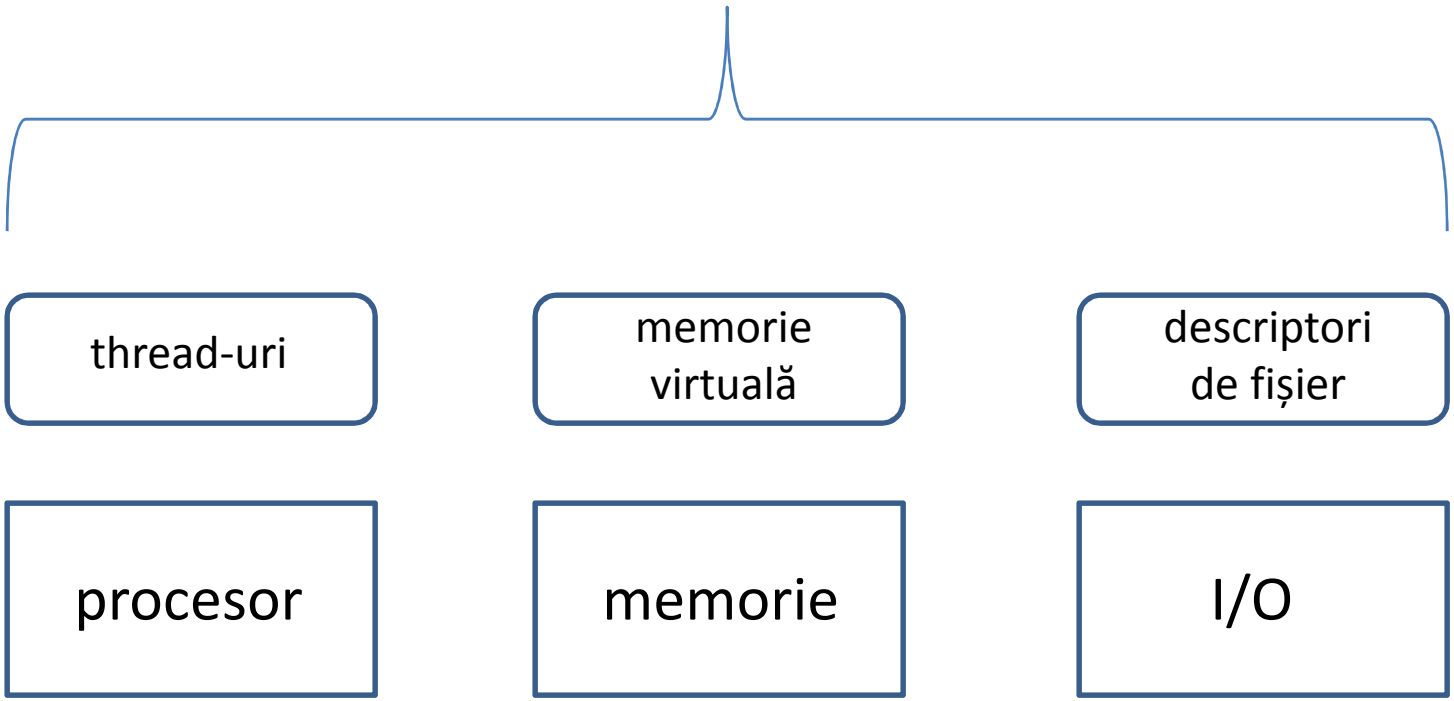
Proces și memorie

- Un proces are o memorie proprie (izolată de alte procese)
- Cod/instrucțiuni și date
- Instrucțiunile sunt aduse din memoria RAM în procesor și executate
  - Spunem că procesul se execută pe procesor

Proces și I/O

- Un proces comunică cu exteriorul: disc, rețea, tastatură, monitor
- Comunicarea se face, de regulă, prin decriptori de fișier
- Un proces are o tabelă de descriptori de fișier
- Un descriptor de fișier referă un fișier, socket, terminal, dispozitiv etc.
- Operațiile cu I/O blochează procesul
  - I/O este mai lent decât procesorul
  - procesul așteaptă încheierea operației

Procesor, memorie, I/O



vom reveni în cursurile 5, 6 și 8

Tipuri de procese

- CPU bound (CPU intensive)
  - rulează des pe procesor
- I/O bound (I/O intensive)
  - rulează rar pe procesor
  - fac operații de I/O -> se blochează

- Interactive (foreground)
  - interacționează cu utilizatorul
- Neinteractive (batch, background)
  - servicii, daemoni

ATRIBUTELE UNUI PROCES

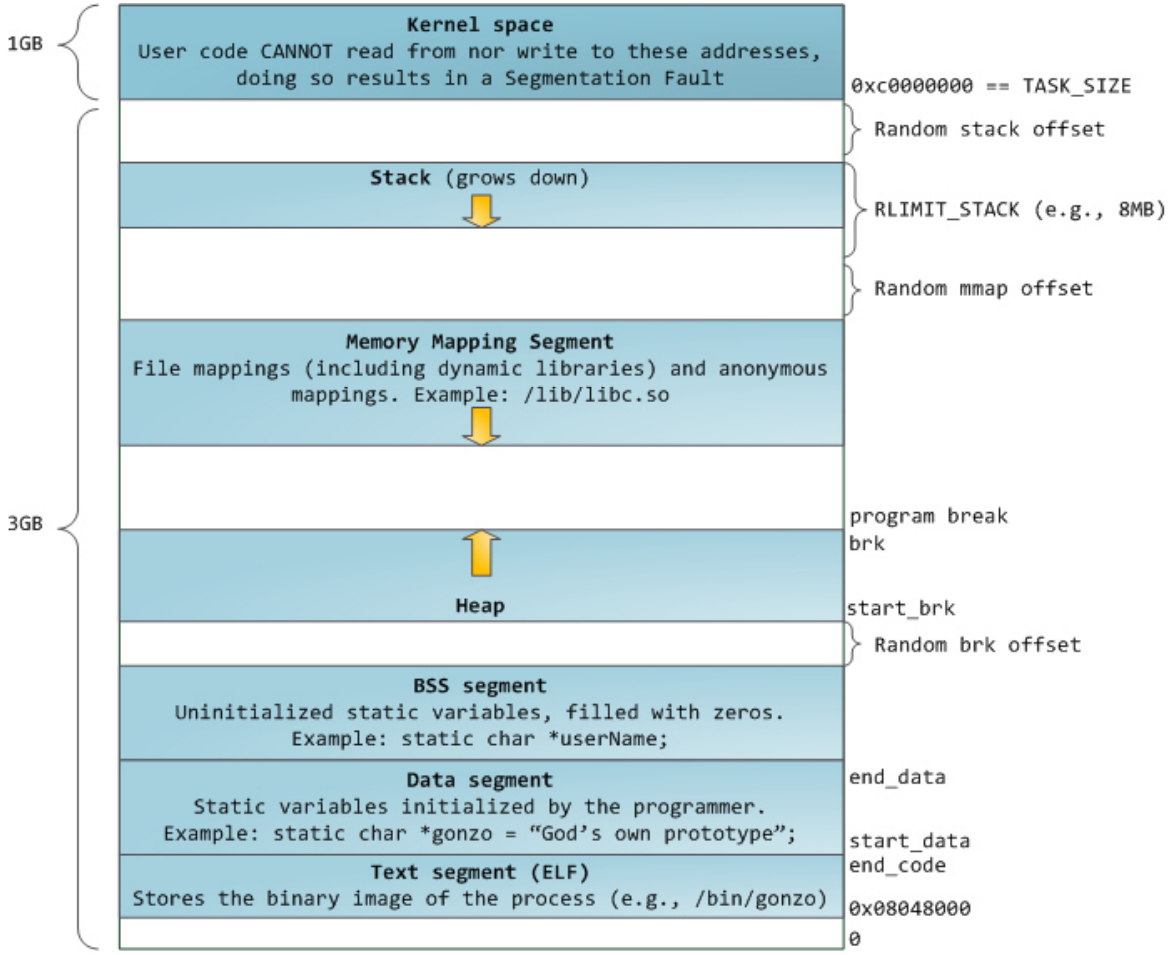
Cum arată un proces la nivelul SO?

- O structură de date
  - PCB (Process Control Block)
  - Descrie un proces la nivelul SO
- Informații legate de resursele folosite
- Un identificator (PID: process identifier)
- Legături cu celelalte structuri
- Informații de securitate, monitorizare, contabilizare

Resursele unui proces

- Timp de rulare pe procesor
- Memorie (cod și date)
- Tabelă de descriptori de fișier
- Unele resurse pot fi partajate cu alte procese

Spațiul (virtual) de adrese



Spațiul (virtual) de adrese (2)

- Fiecare proces are un spațiu (virtual) de adrese
- Asigură izolarea față de alte procese
- Procesul are impresia folosirii exclusive a memoriei
- Toate resursele de memorie (cod, date, biblioteci, stivă) sunt referite prin zone în spațiul virtual de adrese

mai multe în cursul 5

Tabela de descriptori

- Proprie fiecărui proces
- Interfața pentru I/O a unui proces
- Vectori de pointeri către structuri de fișiere deschise
- Structurile pot referi fișiere, socketi, dispozitive speciale (terminale)

Atribute ale unui proces

- PID
- parent PID
- pointeri către resurse
- stare (de rulare, așteptare)
- cuantă de timp de rulare
- contabilizare resurse consumate
- utilizator, grup

PLANIFICAREA PROCESELOR

mai multe în cursul 4

Procese și procesoare

- Pentru a rula un proces are nevoie de un procesor
- Procesorul rulează instrucțiunile procesului
- Procesele într-un sistem sunt mai multe ca procesoarele
- Sistemul de operare asigură accesul echilibrat al proceselor la procesoare

Multitasking

- SO schimbă rapid procesele pe procesoare
- După un timp (cuantă, time slice) un proces este scos de pe procesor și pus altul în loc
  - se spune că “expiră cuanta”
  - acțiunea este numită “schimbare de context” (context switch)
- Cuantă de timp de ordinul milisecundelor
  - se schimbă foarte rapid procesele
  - impresia de rulare simultan
- Un proces poate fi întrerupt în momentul executării unei secvențe

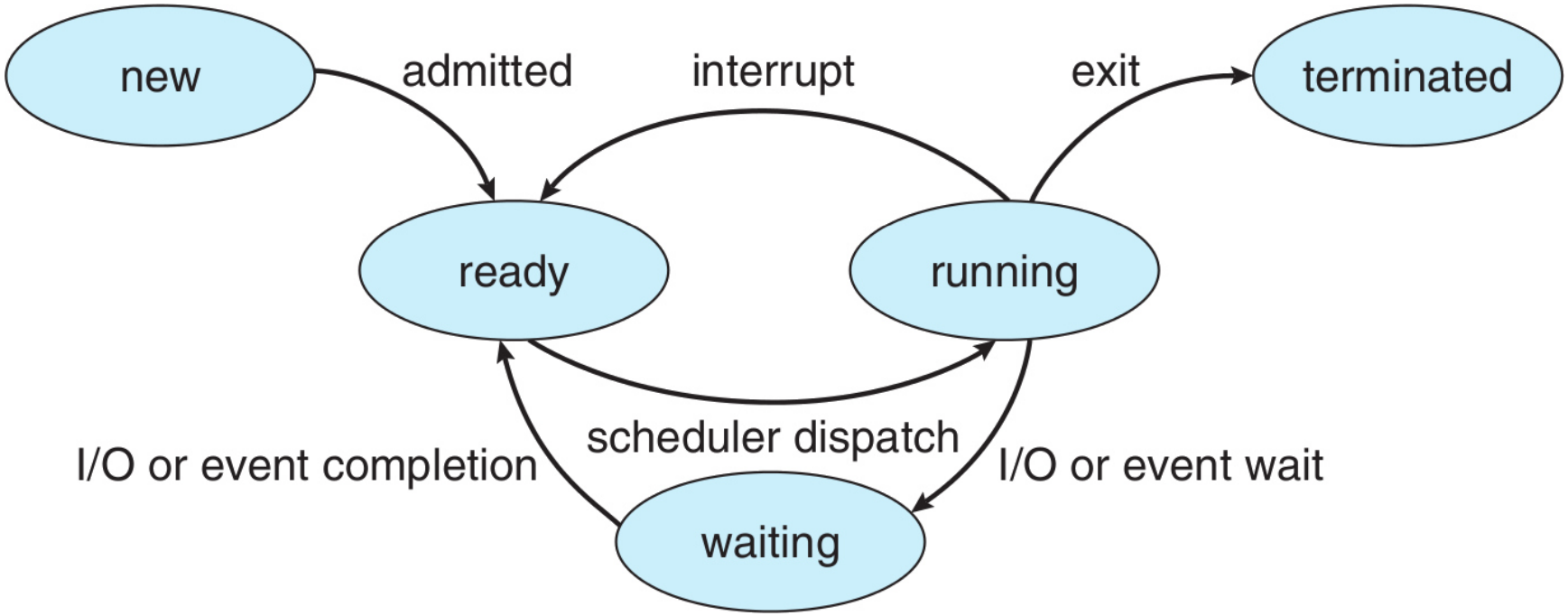
Starea unui proces

- rulare (RUNNING)
  - procesul este pe un procesor și rulează
- așteptare (WAITING)
  - procesul a executat o acțiune blocantă (de exemplu citire I/O) și așteaptă sosirea datelor; nu poate rula
- gata de execuție (READY)
  - procesul poate să ruleze pe procesor
  - pe moment nu are loc pe procesor, e alt proces acolo
- Câte procese se pot găsi în cele trei stări?
- Cum ați asigura gestiunea proceselor în cele trei stări?

Planificarea unui proces (scheduling)

- Un proces este adus din starea READY în starea RUNNING
- Este adus dacă există un procesor liber
- Un procesor este liber dacă procesul de pe procesor l-a eliberat
- Se spune că are loc o schimbare de context (context switch)

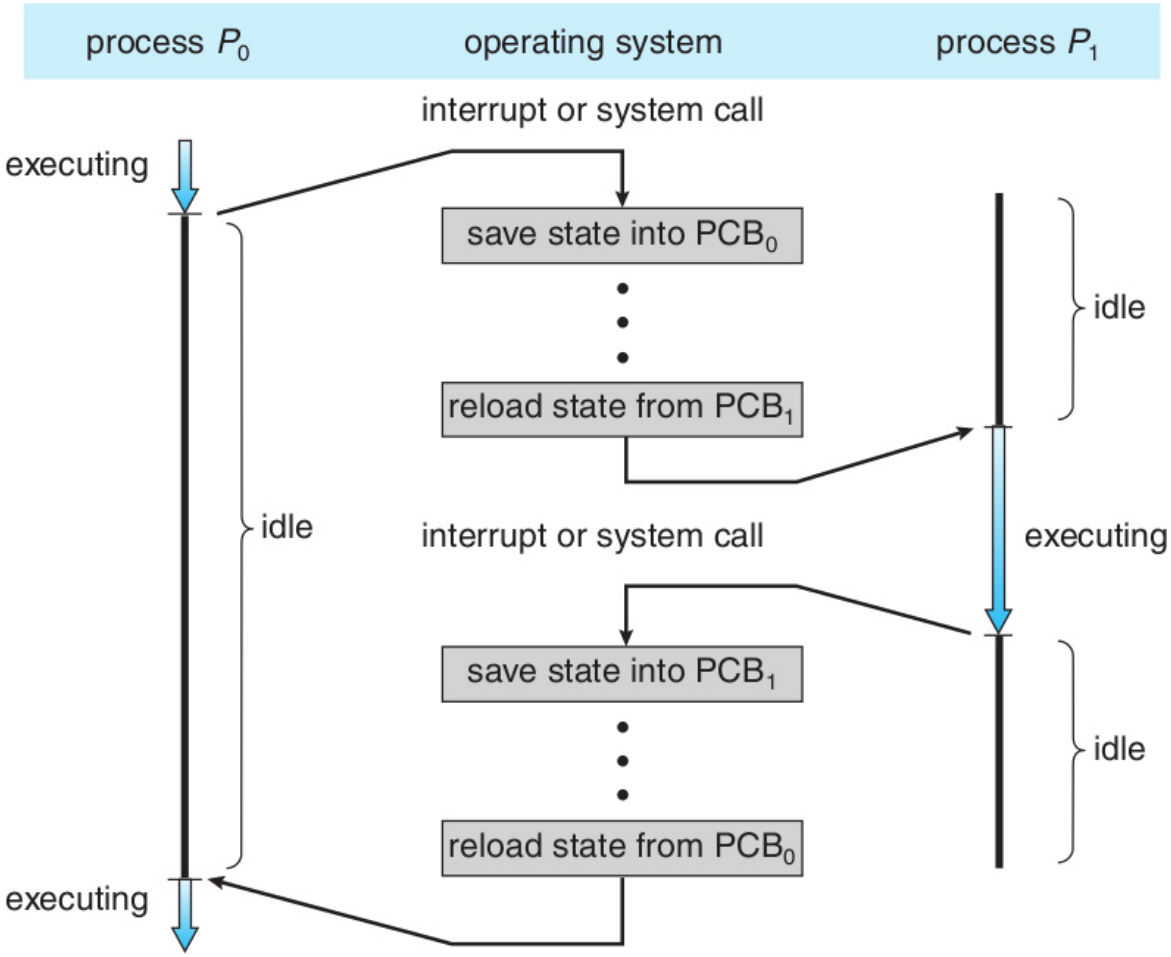
Tranziții între stările unui proces



Tranziții între stările unui proces (2)

- RUNNING -> READY
  - procesului i-a expirat cuanta
  - există un alt proces în starea READY cu prioritate superioară
- RUNNING -> WAITING
  - procesul a executat o operație blocantă
- WAITING -> READY
  - evenimentul așteptat de proces s-a produs
- READY -> RUNNING
  - s-a eliberat un procesor
  - procesul este primul din coada de procese READY

Schimbarea de context



Schimbarea de context (2)

- Un proces este înlocuit pe procesor cu alt proces
- Se salvează procesul/contextul vechi
- Se restaurează procesul/contextul nou
- O schimbare de context înseamnă overhead
  - mai multe schimbări de context: mai mult overhead
  - mai puține schimbări de context: mai puțină interactivitate

Planificatorul de procese

- Componentă a SO
- Responsabilă cu planificarea proceselor
  - asigurarea accesului proceselor la procesoare
  - compromis (trade-off) între echitate și productivitate

mai multe la cursul 4

CREAREA UNUI PROCES

Cum ia naștere un proces?

- Din cadrul unui executabil (program)
- Un alt proces (părinte) creează un proces (copil)
  - noul proces (procesul copil) își populează memoria cu informații din executabil
- Acțiunea se mai cheamă loading, load time
  - realizată de loader

Ierarhia de procese

- Un proces poate crea unul sau mai multe procese copil
- Un proces poate avea un singur proces părinte
- În Unix, procesul init este în vârful ierarhiei de procese
  - init este creat de sistemul de operare la boot
  - init creează apoi procesele de startup
  - procesele de startup creează alte procese etc.

Procese și executabile

- Unul sau mai multe procese iau naștere dintr-un fișier executabil
- Fișierul executabil conține în mod esențial datele și codul viitorului proces
- Procesul are zone de memorie care nu sunt descrise în executabilul aferent
  - stiva
  - heap-ul (malloc)
  - zone pentru bibliotecile dinamice

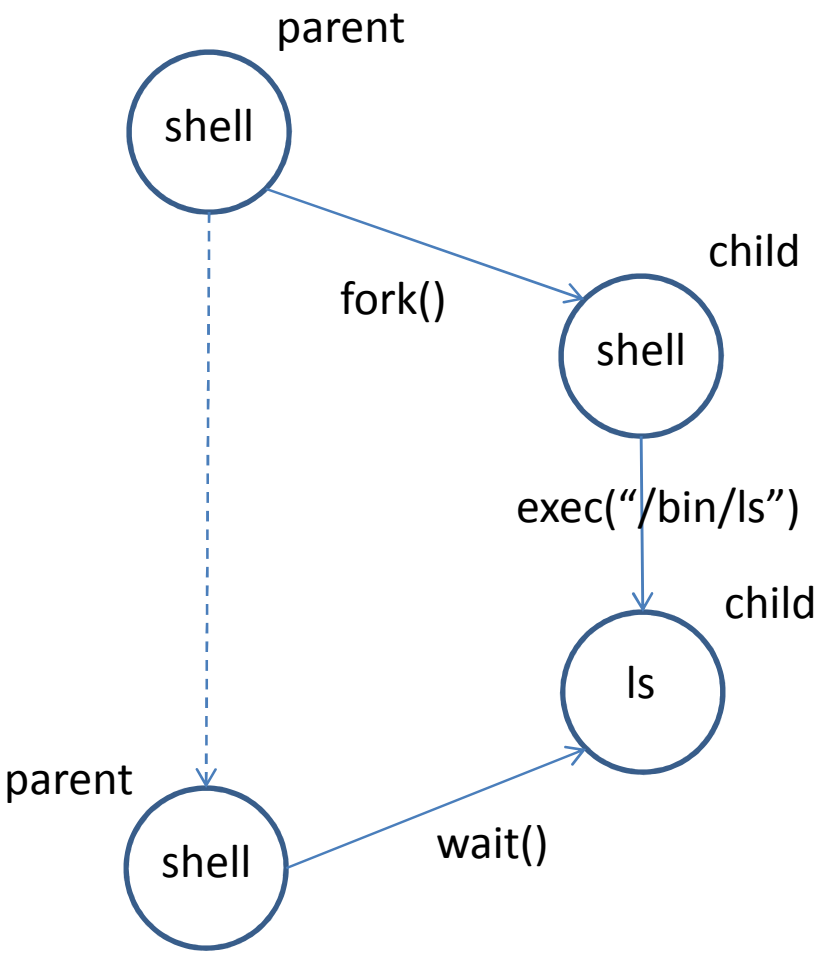
Funcționarea unui shell

- Se scrie la stdin un șir
- Șirul este interpretat de shell într-o cale de executabil (și argumente)
- Procesul shell creează un nou proces (copil)
- Procesul copil “încarcă” (load) datele și codul din executabil
- Procesul copil este planificat de SO
- Părintele procesului copil ste procesul shell

fork și exec

- Separare între crearea unui nou proces și încărcarea datelor dintr-un executabil
- fork: creare nou proces (copil) (aproape identic procesului părinte)
- exec: încărcarea informațiilor dintr-un executabil în memoria procesul copil

fork, exec și shell-ul



fork, exec și redirectare

```
pid = fork();
if (pid == 0) { /* child process */
        fd = open(“a.txt”, O_WRONLY|O_CREAT|O_TRUNC, 0644);
        dup2(fd, STDOUT_FILENO);
        execl(“/bin/ls”,  “/bin/ls”, NULL);
}
```

ALTE OPERAȚII CU PROCESE

Încheierea execuției unui proces

- Procesul își încheie execuția
  - a ajuns la sfârșitul programului
  - a apelat exit()
- Decizia este a procesului
- În final, procesul nu mai există în sistem

Terminarea unui proces

- Sistemul de operare încheie procesul
  - la cererea altui proces (kill)
  - procesul a efectuat o operație nevalidă (segmentation fault)
- Se trimite un semnal (Unix) sau o excepție (Windows)
- În final, procesul nu mai există în sistem

Așteptarea încheierii unui proces

- Sincronizarea acțiunii unor procese
  - faci acțiunea X după acțiunea Y
  - procesul care face acțiunea X așteaptă încheierea execuției procesului care face acțiunea Y
- Se cheamă “waiting for a process”
- Sunt utile informațiile legate de încheierea procesului
  - valoarea de retur, în shell variabila $?

Operatorul shell &

- În mod obișnuit shell-ul așteaptă încheierea proceselor create
- Operatorul &: shell-ul nu mai așteaptă încheierea procesului
  - procesul rulează în background
  - shell-ul controlează stdin

Proces orfan

- Un proces al cărui părinte și-a terminat execuția
- Își pierde legătura în ierarhia de procese
- Este adoptat de un proces dedicat (init pe Unix)

Proces zombie

- Un proces care s-a încheiat dar nu a fost așteptat
- Rămân informații reziduale care vor putea fi folosite de procesul părinte
- Dacă un proces zombie rămâne și orfan, este adoptat de init și apoi este încheiat
- Procesele zombie pe durată mai lungă ocupă (degeaba) resurse ale sistemului

Comunicare interprocese

- Transfer de date între procese
- Partajare de date (de obicei memorie) între procese
- Comunicarea prin fișiere este modul cel mai simplu (și barbar)
- De avut în vedere sincronizarea proceselor
  - un proces citește dacă altul a scris
  - un proces poate scrie dacă altul e pregătit să citească
  - de obicei se folosesc buffere (zone de memorie intermediare)

pipe-uri

- Pipe-uri anonime (spre deosebire de FIFO-uri, pipe-uri cu nume)
- Buffere de comunicare între procese
- Expuse ca o “țeavă” între două procese
  - un capăt de scriere
  - un capăt de citire
  - cele două capete sunt descriptori de fișier
- Se folosesc operațiile de tip read și write
- Procesele trebuie să fie “înrudite”

API DE LUCRU CU PROCESE

Lucrul cu procese în shell

- Se creează prin rularea de comenzi
- Se încheie la încheierea rulării comenzii
- Se termină folosind kill, pkill, Ctrl+c, Ctrl+\
- Shell-ul este procesul părinte al proceselor nou create

Lucrul cu procese în 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 în POSIX

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

pid = wait(&status);
```

Lucrul cu procese în Win32

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

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

Lucrul cu procese în Python

```
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 în Java

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

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

Cuprins

- Rolul proceselor
- Atribute ale unui proces
- Planificarea proceselor
- Crearea unui proces
- Alte operații cu procese
- API pentru lucrul cu procese

Recapitulare

- De ce este nevoie de procese?
- Ce resurse are un proces?
- Câte procese copil sau părinte are un proces?
- Ce este un proces zombie?
- Ce se întâmplă cu un proces orfan?
- Cu ce diferă implementarea pentru rularea “ls” față de rularea “ls &” într-un shell?

Cuvinte cheie

- proces
- procesor
- memorie
- I/O
- CPU bound
- I/O bound
- PCB
- PID
- spațiu virtual de adrese
- multitasking
- stare proces
- READY, RUNNING, WAITING
- schimbare de context
- planificare
- cuantă de timp
- ierarhie de procese
- fork, exec
- terminare proces
- așteptare proces
- zombie
- orfan
- pipe