

Laborator 3

Procese

Sisteme de Operare

3 - 9 Martie 2011

- ▶ program în execuție
- ▶ unitatea primitivă prin care sistemul de operare alocă resurse utilizatorilor
- ▶ caracteristici
 - ▶ spațiu de adrese
 - ▶ unul sau mai multe fire de execuție
- ▶ informațiile asociate procesului (Process Control Block)
 - ▶ tabela de fișiere deschise
 - ▶ handler-ele pentru semnale
 - ▶ directorul curent

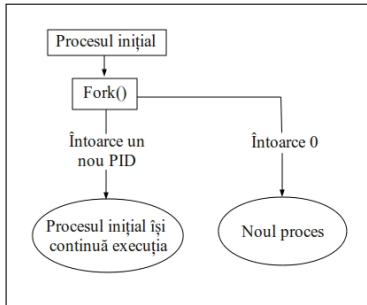
- ▶ creare
- ▶ așteptarea terminării
- ▶ terminare
- ▶ duplicarea descriptorilor de resurse

- ▶ Linux - organizare ierarhică

- ▶ `fork` - **duplică** procesul curent

- ▶ 0, în copil
- ▶ $pid > 0$, în părinte
- ▶ -1, în caz de eroare

- ▶ `exec` - **înlocuiește** imaginea procesului



- ▶ Windows - organizare neierarhică

- ▶ `CreateProcess` - îmbină cele două operații de pe Linux

▶ Linux

- ▶ `waitpid, wait`
 - ▶ suspendă execuția procesului apelant până când procesul (procecele) specificat în argumente fie s-au terminat, fie au fost oprite (`SIGSTOP`)
- ▶ `WIFEXITED, WEXITSTATUS ...`
 - ▶ obțin modul și codul de ieșire ale procesului, examinând status, întors de `waitpid`

▶ Windows

- ▶ `WaitForSingleObject, WaitForMultipleObjects`
 - ▶ suspendă execuția procesului curent până când unul sau mai multe alte procese se termină
- ▶ `GetExitCode`
 - ▶ determină codul de eroare cu care s-a terminat un anumit proces

▶ Linux

▶ exit

- ▶ încheie execuția procesului curent
- ▶ toți descriptorii de fișier ai procesului sunt închisi
- ▶ copiii procesului sunt "înfiți" de init
- ▶ părintelui procesului îi e trimis un semnal SIGCHLD
- ▶ va scrie bufferelor streamurilor deschise și le va închide

▶ Windows

▶ ExitProcess

- ▶ încheie execuția procesului curent

▶ TerminateProcess

- ▶ încheie execuția altui proces
- ▶ **Nu** este recomandată

- ▶ Linux
 - ▶ dup, dup2
 - ▶ descriptorii din părinte se moștenesc, implicit, în copil

- ▶ Windows
 - ▶ descriptorii ce indică fișierele către care se face redirectarea trebuie să poată fi moșteniți în procesul creat
 - ▶ membrul `bInheritHandle` al structurii `SECURITY_ATTRIBUTES` pasate lui `CreateFile` trebuie să fie `TRUE`
 - ▶ pentru moștenirea descriptorilor
 - ▶ parametrul `bInheritHandle` din `CreateProcess` trebuie să fie `TRUE`
 - ▶ la crearea procesului, trebuie populată structura `STARTUPINFO`
 - ▶ setarea membrilor `hStdInput`, `hStdOutput`, `hStdError` la descriptorii corespunzători
 - ▶ membrul `dwFlags` trebuie setat la `STARTF_USESTDHANDLES`

▶ Linux

- ▶ `int main(int argc, char **argv, char **environ)`
 - ▶ parametrul `environ` e un vector de șiruri de caractere de forma `VARIABILĂ = VALOARE`
- ▶ `getenv, setenv`
 - ▶ obține/setează valoarea unei variabile de mediu
- ▶ `unsetenv`
 - ▶ înlătură o variabilă de mediu

▶ Windows

- ▶ `GetEnvironmentVariable, SetEnvironmentVariable`
- ▶ setarea unei variabile cu valoarea `NULL` înlătură acea variabilă

- ▶ mecanisme de comunicare între procese, ce oferă acces de tip FIFO
- ▶ sistemele de operare garantează sincronizarea între operațiile de citire și de scriere la cele două capete
- ▶ două tipuri
 - ▶ **anonime**
 - ▶ pot fi folosite doar între procese înrudite
 - ▶ există doar în prezența proceselor care dețin descriptori către ele
 - ▶ **cu nume**
 - ▶ pot fi folosite între oricare două procese
 - ▶ există fizic - sunt reprezentate de fișiere speciale

Linux

- ▶ pipe
- ▶ read, write
- ▶ close

Windows

- ▶ CreatePipe
- ▶ ReadFile, WriteFile
- ▶ CloseHandle

Atenție!

- ▶ **Linux:** Când se utilizează `fork`, descriptorii sunt duplicați => numărul necesar de închideri se vor dubla. Închiderea parțială a descriptorilor conduce la blocaje în `read`.
- ▶ **Windows:** Valorile descriptorilor nu sunt direct vizibile în procesul copil și trebuie făcute cunoscute printr-o metoda alternativă.

- ▶ moduri de deschidere
 - ▶ blocant
 - ▶ neblocant

- ▶ Linux
 - ▶ mkfifo

- ▶ Windows
 - ▶ moduri de comunicare
 - ▶ flux de octeți
 - ▶ flux de mesaje

Server

- ▶ CreateNamedPipe
- ▶ ConnectNamedPipe

Client

- ▶ CreateFile
- ▶ CallNamedPipe

- ▶ Care este primul argument (`argv[0]`) al programului 'fun' dacă se rulează astfel: `./fun so lab 3`
- ▶ Ce va afișa apelul `getppid()` efectuat de un proces copil al cărui părinte a apelat `exit()` ?
- ▶ Câte procese noi se vor crea în urma execuției codului următor: `if (fork()) fork();`
- ▶ Ce se întâmplă dacă un program execută următorul cod: `execvp(argv[0], &argv[0])`