SISTEME DE OPERARE

- Laborator 9 -

PROCESE UNIX

1. PROCESE UNIX

* *proces* = *program aflat în execuţie*
* nucleul (*kernel*) sistemului de operare întreţine permanent o tabelă cu procese
* examinarea proceselor:

ps -e

ps -f -p 1

ps -F -u dbota

2. FUNCŢIA fork()

* prototip:

|  |
| --- |
| #include <unistd.h>  **pid\_t fork(void);** |

* crează un nou proces prin duplicarea procesului apelant (procesul părinte)
* noul proces este denumit proces copil şi este o copie aproape exactă a procesului părinte
* cele două procese îşi continuă execuţia cu instrucţiunea care urmează apelului **fork()**
* returnează:
  + valoarea 0 - în procesul copil
  + identificatorul procesului copil (child PID) - în procesul părinte
  + valoarea -1 - dacă apelul a eşuat
* apelul funcţiei **fork()** eşuează în următoarele situaţii:
  + nu există spaţiu de memorie suficient pentru duplicarea procesului părinte
  + numărul total de procese depăşeşte limita maximă admisă
* exemple: fork\_1.c, fork\_2.c, fork\_3.c

3. FUNCŢIILE wait(), waitpid()

* prototipuri:

|  |
| --- |
| #include <sys/types.h> #include <sys/wait.h>  **pid\_t wait(int** *\*status***);**  **pid\_t waitpid(pid\_t***pid***, int** *\*status***, int***options***);** |

* **wait()** suspendă execuţia procesului apelant până la terminarea unui proces copil
* apelul **wait(&status)** este echivalent cu **waitpid(-1, &status, 0)**
* **waitpid()** suspendă execuţia procesului apelant până la apariţia unuia dintre următoarele evenimente:
  + - procesul fiu specificat prin argumentul **pid** şi-a terminat execuţia
    - procesul fiu specificat prin argumentul **pid** a fost oprit printr-un semnal
    - procesul fiu specificat prin argumentul **pid** a fost repornit printr-un semnal
* semnificaţia valorilor argumentului pid:

|  |  |
| --- | --- |
| pid | Semnificaţie |
| < -1 | Se aşteaptă terminarea tuturor proceselor copil al căror identificator de grup (GID) este egal cu valoarea absolută a parametrului pid |
| -1 | Se aşteaptă terminarea tuturor proceselor copil |
| 0 | Se aşteaptă terminarea tuturor proceselor copil al căror identificator de grup (GID) este egal cu GID-ul procesului părinte |
| > 0 | Se aşteaptă terminarea procesului cu PID-ul specificat prin parametrul pid |

* exemple: fork\_4.c, fork\_5.c

4. FUNCŢIA signal()

* prototip:

|  |
| --- |
| #include <signal.h>  **sighandler\_t signal(int** *signum***, sighandler\_t** *handler***);** |

* stabileşte modul de acţiune la apariţia unui semnal
* semnale:

man 7 signal

* dacă semnalul *signum* poate fi livrat unui proces, atunci acesta poate să aleagă:
  + să ignore semnalul - SIG\_IGN
  + să-l trateze în mod implicit - SIG\_DFL
  + să specifice o funcţie care defineşte acţiunile care se execută la apariţia semnalului
* prevenirea apariţiei proceselor de tip „zombie”:

signal(SIGCHLD, SIG\_IGN)

* exemplu: fork\_7.c

5. FUNCŢIA kill()

* prototip:

|  |
| --- |
| #include <sys/types.h>  #include <signal.h>  **int kill(pid\_t** *pid***, int** *sig***);** |

* permite livrarea unui semnal unui proces sau grup de procese
* semnificaţia valorilor argumentului pid:

|  |  |
| --- | --- |
| pid | Semnificaţie |
| > 0 | Semnalul este livrat procesului cu PID-ul specificat prin parametrul pid |
| 0 | Semnalul este livrat tuturor proceselor al căror identificator de grup (GID) este egal cu GID-ul procesului apelant |
| -1 | Semnalul este livrat tuturor proceselor pentru care procesul apelant are dreptul de a livra semnale (cu excepţia procesului init) |
| < -1 | Semnalul este livrat tuturor al căror identificator de grup (GID) este egal cu valoarea absolută a parametrului pid |

* semnale:

man 7 signal

REFERINȚE:

* Tutorial IPC: https://beej.us/guide/bgipc/html/single/bgipc.html