SO

Functionalitățile SO.

1) interfață cu utilizatorul
2) gestionarea perifericelor
3) Gestionarea fișierelor
4) Apeluri sistem

cond nec pt a fi SO

5) Gestionarea proceselor - multitasking
6) Gestionarea și protecția memoriei
7) Gestiunea utilizatorilor
8) Lucru în Rețea.

Terminale

2 tipuri → alfanumerice (mod linie de comandă)
→ grafice (reprez. prin pixeli)

Ceasuri

→ circuite programabile
→ întreruperi aduc pc-ul la dispoziție pt a putea da controlul altui proces

· Avem în ordine de execuție $P_0, P_1 \dots P_n$

P_K se execută

Vine întreruperea:

→ se salvează contextul
→ alegere proces care trebuie continuat și cât timp îi alo
→ refacere context proces ales
→ programare întrerupere
→ salt la proces (mutare PC)

Caracteristici procese:

→ context
→ identificator unic (PID)
→ identificator unic părinte (PPID)
→ stare proces
→ Ready → în tabelă, așteaptă să fie continuat
→ Running → momentul în care este ales
→ User mode
→ Kernel mode
→ SLEEPING
→ STOPPED
→ Zombie

- directorul curent

-> variabile de mediu și valorile lor;

APELURI SISTEM

->testare codul de retur.
->start și kill procese UNIX
->terminarea procesului void exit(int k)

FORK:

cod de retur -1 -> ~~eroare~~ (Tabela de procese plină)
procesul copil primește 0 => tatăl primește cod de retur > 0 = PID-ul fiu

int wait(int * stare) -> -1 înseamnă că nu are fii

<--! int stare;
wait(& stare);

-->.

int waitpid(int pid, int *stare, int optiuni)

optiuni: WNOHANG | WUNTRACED
↳ fără asta așteaptă doar după zombie

Apeluri din familia exec:

codul lui de retur poate fi doar -1
cel mai precum cod de eroare => nu există fișierul executabil

int execl(char *exe, char arg0, char arg1 ..., NULL
↳ cale explicită

int execlp(char *exe, char arg0 ... NULL)
↳ tine cont de path.

Comunicare între procese:

->mod normal (complet nerec)
->fișiere speciale (pipe și socket)
->IPC System V: creatori de semafoare, ~~ipc~~ mem partajată, cozi de mesaje

->Semnale;

Deadlock

P_1, P_2 procese
R_1, R_2 resurse

$P_1 \leftarrow R_1$
$P_2 \leftarrow R_2$
$P_1 \leftarrow - R_2$
$P_2 \leftarrow - R_1$

Se alege unul dintre procese victimă, apelul ~~de~~ sistem în care este blocat procesul victimă iese cu cod de retur -1 și acesta ia valoarea EDEADLOCK

Procesul victimă se alege în funcție de cât a costat până la momentul respectiv, și cât ar costa să se reia operațiu

Probleme de concurență

Problema producător - consumator
Problema filosofilor
Problema cititori - scriitori
Problema frizerului somnoros

Planificarea proceselor:

ROUND-ROBIN (Princ. manivelei) - UNIX
- se impart procesele după priorități 0, 1, 2, 3
- se tratează procesele cu ~~prioritatea 00~~ în fct de prioritate
  0 - cel mai prioritar

Preemptive -> se bazează pe evenim anterioare

Gestiunea memoriei

FIFO -> Nerecomandat

best fit -> caută zona în care se încadrează procesul așa că se piardă cât mai puțină memorie

LRU - Least Recently Used

Periferice:

- de intrare
- de ieșire
- bidirecționale
  - Human readable (tastatură)
  - Machine readable (Hdd)

mode de funcționare
- caracter cu caracter
- bloc