fishier - écuiture ouverture

int to = open (dremin, flag, mode)

lecture : écriture : les & :

ne pas oublier de Jenner: close (td)

read (td, but, taille - but)

retourne le noumbre écuit étaille-but write (dd, bub, n)

SEEK. SET (seek (td, ollset, whence) dicalage of

SEEK - CUR SEEK - END

Insend

break; break; break; break; break; break; break; switch (a.t._code & 2_IRM) (
case \$_IRM, continue (block);

ca //avec lstat test_lien => lien symbolique
perror("stat");
exit(1); orintf("n° inoeud :%ld \n", st.st_ino); printf("Type de fichier : "); res = lstat(argv[1], &st);

Random

Srand (+ime (NULL) ; rand () 7. 11Ax;

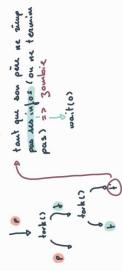
Repertour

Se diplace dans la dossiers

if (stromp(direntp->d_name,".")==0 | stromp(direntp->d_dame,".")==0 | continue; switch (direntp->d_type) [case DI_REG: //pour compter les fichiers normau countf(char *path) {
 DIR *df_per * NUL;
 struct dirent dirent)
 consert(path : NUL);
 consert(path : NUL);
 if (df_ptr = opendir(path)) == NUL) return 0; int count=0; while((direntp = readdir(dir_ptr))) closedir(dir_ptr);
return count;

Processus

· Comment oren n jib



Ping - Pong

int main(void)!

**grant(stoidst., handler);

ptd = 10rt();

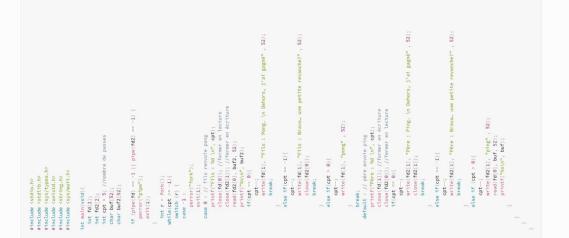
ripid = 10, quand, if actoil 5c on modal; c le pro comm

ripid = 20, squand if actoil 5c on modal; c le pro comm

ripid = 10, squand or j.

ripid = 10, squand or j.

ripid = 10, squand or j. void handlerf(int sig){
if(sig == SIGALRM){
 kill(pid, SIGSR1);
 signal(SIGUSR1, handlerf); void handler(int sig){
if(sif == SIGALRM)!
kill(pid, SIGSR1);
signal(SIGUSR1, handler); else(
 printf("Pong \n");
 signal(SIGALRM, handler);
 alarm(2); printf("Ping \n");
signal(SIGALRM, handler);
alarm(2); #include <stdio.h> #include <signal.h> pid_t pid;





- manipulable presque comme un fichier ordinaire descripteur, read, write...
 la lecture est destructrice: tout octet lu est consommé et retiré du tube
 flot continu de caractères: pas de séparation entre 2 écritures successives
- fonctionnement de type fifo, unidirectionnel: un tube a une extrémité en écriture et une
- capacité limitée (donc notion de tube plein)
- par défaut, les opérations sur les tubes sont bloquantes
 un tube est auto-synchronisant : impossible de lire un caractère avant qu'il ne soit écrit !

nt pipe(int pipefd[2]);

- crée et ouvre un tube anonyme donc alloue :

- s stocke ces descripteurs dans pipefd: l'ecture dans pipefd(0), écriture dans pipefd(1), er renvoire lo en cas de succès, -1 en cas déchec (si la table de descripteurs du processus on la table des ouvertures de l'others est pielne)

e tube créé n'est accessible que via ces 2 descripteurs - comme il n'a pas de nom, on ne » seuls les descendants du processus qui a créé un tube anonyme peuvent donc y peut pas le réouvrir avec open.

 si le tube n'est pas vide et contient taille octets, nb_lus = min(taille, TAILLE_BUF) octets accéder, en héritant des descripteurs.

- sont extraits et copiés dans buf
 si le tube est vide, le comportement dépend du nombre d'écrivains (i.e. de descripteurs
 - en écriture sur le tube) : renvoie nb_lus = 0 si le nombre d'écrivains est nul
- sinon, par défaut la lecture est bloquante : le processus est mis en sommeil jusqu'à ce que quelque chose change (contenu du tube ou nombre d'écrivains)

e caractère bloquant permet la synchronisation d'un lecteur sur un écrivain... mais peut également provoquer des auto- ou interblocages

Tubes nommés

de même nature que les tubes anonymes, mais avec une existence dans le SGF :

- accessibles par des processus non nécessai
- persistants (enfin... pas leur contenu)

</> </> Création

Suppression avec unlink(l), renommage avec rename(l), changement des droits avec chmod(l)... comme les autres entrées de répertoires

Parfois le comportement bloquant par défaut des tubes n'est pas adapté ; on peut le

- it is towerture dun tube nommé, avec le flag O_NONBLOCK
 is sous Linux jà a tredelinqueventure d'unt lube anonyme par
 int pipe2(int pipe2(iz), int flags), avec le flag O_NONBLOCK
 après louverture, en modifiant les flags du descripteur avec
 int fent(int de, int cmd, .../* arg */)
- comportentent d'une ouverture non bloquante :

 en lecture, eller réusalt immédiatement;

 en fecture, elle réusalt soulement en présence d'un lecteur; sinon elle échoue, avec enne=ENXIO

```
· App' à table
```

Lo tous hofils arrivent

La fila-n arrive à n appel

```
D at pen
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   1 to ....
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   int main(void) {
    signal(SIGUSR1, handler);
    pid = fork();
    if(pid = 0){
        signal(SIGUSR1, handlerf);
        ktll(getppid, SIGUSR1);
                                                                                                                                                                                                                                                                               void handlerf(int sig){
if(sig == SIGALRN){
   kil(pid, SIGSR1);
   signal(SIGUSR1, handlerf);
                                                                                                                                                                                                                                                                                                                                                             else{
   printf("Ping \n");
   signal(SIGALRM, handler);
   alarm(2);
                                                                           void handler(int sig){
if(sif == SIGALRM);
kill(pid, SIGSRl);
signal(SIGUSRL, handler);
                                                                                                                                                         else{
  printf("Pong \n");
  signal(SIGALRW, handler);
  alarm(2);
#include <stdio.h>
                                              pid_t pid;
```

l'un à la suit : fous les chier a changer signal, pus methre en pours, puis envoic du signal

```
terminalson du processus
- SIGINT, SIGTERM, SIGKILL...
terminalson + génération d'un fichier core
- SIGQUIT, SIGSEGV...
                                                                           signal ignoré
- SIGCHLD, SIGWINCH
suspension du processus
- SIGSTOP, SIGTSTP
reprise du processus
```

Principaux signaux

```
    problèmes matériels: SIGBUS, SIGSEGV, SIGILL, SIGFPE 2
```

• événements « externes » : SIGCHLD 3 , SIGPIPE 1

- SIGTERM, SIGKILL pour terminer 1
 SIGTERM, SIGTSTIP pour terminer 4
 SIGCONT pour reprendre 5
 SIGCONT pour reprendre 5
 SIGCONT pour reprendre 5
 SIGCONT pour reprendre 5
 SIGCONT pour reprendre 6
 SIGCONT pour reprendre 7
 SIGCONT pour reprendre 7
 SIGCONT pour reprendre 6
 SIGCONT pour reprendre 7
 SI
- événements liés au terminal :

- SIGHUP : déconnexion 1
 SIGNT 1 , SIGTSTP 4 , SIGQUIT 2 : ctrl-C, ctrl-Z, ctrl-\
 SIGTNT 1, SIGTSTP 4 : tentative de lecture/écriture par un processus à l'arrière-plan
 SIGTTIN, SIGTTOU 4 : tentative de lecture/écriture par un processus à l'arrière-plan
 - SIGWINCH 3 : redimension
- auto-notification: SIGABRT 2, SIGALRM 1

sans signification prédéfinie : SIGUSR1, SIGUSR2 1

power (): permet de bloquer un processus jusqu'à la recep^e d'un signal

```
//appel bloquant
scanf("entrez 1 : %d", &n);
printf("aon fils %d est mort ovec %u: \n", (int) fils ,erreur);
                                                                                                                                            if( pid == -1) fprintf(stder, 'impossible de creer le fils %d \n', errno);
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          fprintf(stdout, "PID-%d:", getpid());
printf("s suis le pere de %d", pid );
printf("..et je suis le fils du shell %d \n",getppid());
                                                                                                                                                                                                            if ( pid == 0 { { //Fils
print(Pilo4d: ",getpid());
print(")s auts le fils de %d \n", getppid() );
//oppi bloquart
sconf("valaur pour le fils %d \n",&n);
                                                                                                 pid - fork(); //essai de creation fils
int main( int mbargs , char ** args ) {
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        //attente du fils
fils = wait(&erreur);
                                                                                                                                                                                                                                                                                                                                                                                                                                                  else { //pere
int n, erreur;
pid_t fils;
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    return EXIT_SUCCESS;
                                                                                                                                                                                                                                                                                                                                                                                exit(1);
                                                      pid_t pid ;
```