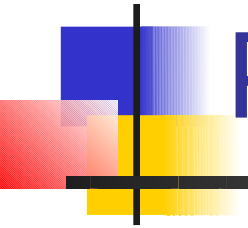


Processus, Tube et Shell

Processus

- Création de processus
  - Effectué par clonage d'un processus existant
    - =>Duplication du processus
  - Appel système fork()
    - Retourne 0 pour le processus fils
    - Retourne le pid du fils pour le processus père
    - Retourne -1 si erreur

```
#include <unistd.h>

pid_t fork(void)
```

```
r = fork();
if (r==-1) …          /* erreur */
else if (r==0) …      /* code du fils */
else …                /* code du père */
```

Fork

- Combien de processus sont créés

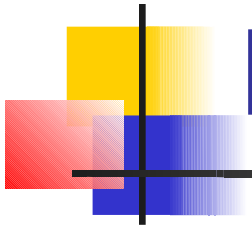
```
fork();
fork();
fork();
```

```
for (i=0; i<3;i++){
   fork();
}
```

- Donner les diverses traces possibles

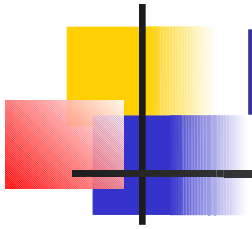
```
int i = 0;
switch((i=fork()) {
            case -1 : perror("fork"); break;
            case  0 : i++; printf("fils I :%d",i); break;
            default : printf("pere I :%d",i);

}
```



Primitive de recouvrement

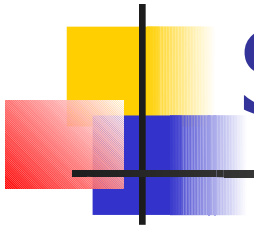
- Rappel : définition d'une fonction main
  - int main(int argc, char *argv[]);
- Appel execvp
  - Remplacer l'image d'un processus
  - int execvp(const char *ref, const char *argv[])
    - ref : nom du fichier à charger
    - argv : paramètres de la commande à charger
  - execvp appelle main(argc, argv) sur la commande à lancer



Exemple

```
char  * argv[3];

argv[0] = "ls ";
argv[1] = "-al ";
argv[2] = 0;
execvp("ls", argv);
```



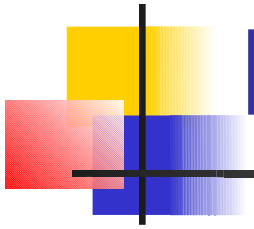
Synchronisation Père/Fils

- Synchronisation d'un processus père sur la terminaison de ses descendants.
  - pid_t wait(int *status)
    - Le père attend la mort d'un de ses fils
      - pid_t : pid du fils mort ou -1 si pas de fils
      - status : info sur la terminaison du fils
  - pid_t waitpid(pid_t pid, int *status, int option)
    - Le père attend la mort d'un fils particulier
    - Option : non bloquant … voir man

Exemple

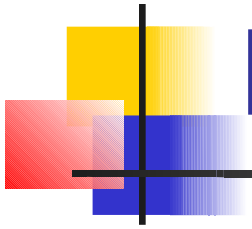
```
#include <sys/types.h>
#include <sys/wait.h>

main(){
  int spid, status;
  switch(spid = fork()){
            case -1 : perror(…); exit(-1);
            case 0  :  // code du fils
                       break;
            default : // le pere attend la terminaison du fils
                       if (waitpid(spid,&status,0)==-1) {perror(…);exit(-1);}
                       …
  }
}
```



Entrées/Sorties (1/2)

- Tout fichier est désigné par l'intermédiaire d'un descripteur
  - 0, 1 et 2 correspondent à l'entrée standard, à la sortie standard et à la sortie erreur standard
  - Le numéro de descripteur d'un fichier est fourni comme retour de la primitive open

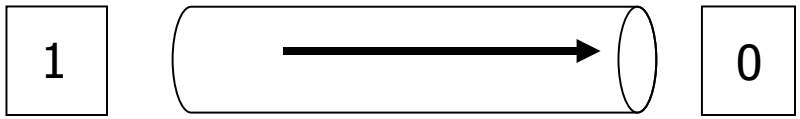


Entrées/Sorties (2/2)

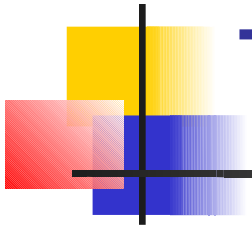
- Primitives de base
  - int open(const char *ref, int mode);
    - O_RDONLY, O_WRONLY, O_RDWR …
  - int creat(const char *ref,mode_t droit);
  - int close(int desc)
  - ssize_t read(int desc, void *ptr,size_t nb_octet);
  - ssize_t write(int desc, void *ptr, size_t nb_octet);

Tube

- Mécanisme de communication entre processus
  - Structure fifo
  - Capacité limitée
  - Synchronisation producteur/consommateur

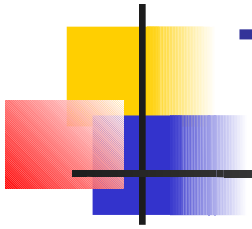


tube



Tube

- Primitive de création d'un tube (<unistd.h>)
  - int pipe(int p[2])
  - Crée 2 descripteurs de fichier
    - p[0] permet de lire dans le tube
    - p[1] permet d'écrire dans le tube
  - Retour
    - 0 : création effectuée
    - -1 : erreur

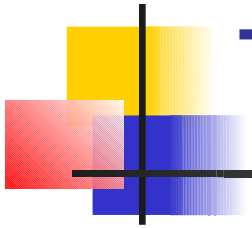


Tube

- Utilise les primitives des fichiers
  - read, write, close

int read(int desc, void * buf, int nboctet);
desc : descripteur en lecture
buf : zone de stockage des octets lus
nboctet : nb octet à lire
Retour : int nb octet effectivement lus

int write(int desc, void * buf, int nboctet);
desc : descripteur en écriture
buf : data à écrire
nboctet : nb octet à écrire
Retour : int nb octet effectivement écrit



Tube

- Duplication de descripteurs
  - dup(int desc); dup2(int desc_src, int desc_dest);
  - Utilisé pour rediriger E/S standard

```
#include <stdio.h>
#include <unistd.h>
int p[2];
/* redirection entrée standard vers le tube */
pipe(p);
…
close(STDIN_FILENO); // ferme entrée standard
dup(p[0]);           // duplique p[0] sur le 1er descripteur libre (i.e. 0)
close(p[0]);         // on libère p[0]
…
```

```
pipe(p);
dup2(p[0],STDIN_FILENO);
close(p[0]);
```

TP Shell

- Travail à réaliser
  - Interpréteur de commandes
- Commandes
  - Externes : création d'un nouveau processus
  - Internes : exécution en interne
- Caractères spéciaux
  - Redirection : '**>**', '**<**'
    - ls –al > toto
  - Lancement concurrent : '**|**'
    - ps -e| wc -l
  - …

Exemples :

ls –al #externe
cd #interne

Exemples :

ls –al > toto
ps -e| wc –l
ps -e| wc –l > toto

Fichiers fournis

- Sujet
- Analyseur de ligne de commande
  - readcmd.h
  - readcmd.c
  - test.c (petit essai de readcmd)

```
struct cmdline {
        char *err;
        char *in;
        char *out
        char ***seq
};
```

```
struct cmdline *cmd;
cmd = readcmd();
```