Rappels et compléments UNIX (II)

Gestion des ressources et appels système

LaBRI, UMR CNRS 1308
Université Bordeaux 1

Serge Chaumette
Serge.Chaumette@labri.fr

Objectifs de ce cours

- Objectifs
  - Assurer un niveau homogène
  - Assurer un niveau minimum garanti
  - Maîtriser les principes de base du système
- Évaluation
  - Exercices
    - Au fur à mesure des manipulations
    - Au fur à mesure des autres cours

Mise en garde

- Tous les exemples présentés ici sont écrits en utilisant le langage C.
- On se place dans un environnement de type UNIX/Linux et de légères variantes peuvent être constatées d'une implémentation à une autre.

Contenu

- Gestion des utilisateurs
- Gestion des fichiers
- Gestion des processus

Gestion des Utilisateurs

- Identification
  - Intra système
    - Mot de passe, *uid* UNIX
  - Inter systèmes
    - Notion d'annuaire
- Droits d'accès
  - Qualitatif
  - Quantitatif
    - Quotas, facturation

Exemple de structure d'information

```
struct passwd {
        char *pw_name;    /* user's login name */
        char *pw_passwd;  /* no longer used */
        uid_t pw_uid;      /* user's uid */
        gid_t pw_gid;      /* user's gid */
        char *pw_age;     /* not used */
        char *pw_comment; /* not used */
        char *pw_gecos;   /* typically user's full name */
        char *pw_dir;     /* user's home dir */
        char *pw_shell;   /* user's login shell */
      };
```

Notion d'annuaire

- Un annuaire est un service permettant de répertorier des informations (utilisateurs, services, etc.)
- Exemples
  - Fichier /etc/passwd UNIX
  - NIS (ex. YP)
  - LDAP

Nouvelles problématiques

- Mobilité des utilisateurs
- Multiplication des points d'accès

Gestion des Fichiers

- Fonctionnalités
  - organisation
  - accès
    - différents modes
  - interrogation

Répertoires

- Ouverture/fermeture
  - DIR *opendir(const char *nom)
  - int closedir(DIR *dir)
- Positionnement
  - void rewinddir(DIR *dir)
  - void seekdir(DIR *dir, off_t offset)
  - off_t telldir(DIR *dir)

Répertoires (suite)

- Lecture
  - struct dirent *readdir(DIR *dirp)

```
typedef struct dirent {
        ino_t          d_ino;          /* "inode number" of entry */
        off_t          d_off;          /* offset of disk directory entry */
        unsigned short  d_reclen;      /* length of this record */
        char           d_name[1];      /* name of file */
} dirent_t;
```


Interrogation

- Exemple

```
...
struct stat s;
...
stat("toto/titi", &s);
if (S_ISDIR(s.st_mode))
...
```

Exercice

- Écrire un équivalent rudimentaire de la commande UNIX ls, c'est à dire qui se comporte comme dans l'exemple :

```
$ ls
a.out
toto.txt
tmp/
alpha.o
$
```


Fichiers

- Ouverture/fermeture
  - open
  - close
- Positionnement
  - lseek
- Lecture/écriture
  - read
  - write
- Divers
  - stat, chmod, chown

Select

```
int select (int maxfd,
            fd_set *readfs,
            fd_set *writefs,
            fd_set *exceptfs,
            struct timeval *timeout) ;

FD_ZERO(fd_set *fdset) ;
FD_SET(int fd, fd_set *fdset) ;
FD_CLR(int fd, fd_set *fdset);
FD_ISSET(int fd, fd_set *fdset) ;

struct timeval {
            long tv_sec;
            long tv_usec;
 } ;
```


Nouvelles problématiques

- Fichiers distribués
  - NFS, AMD
- Entrées-sorties parallèles
  - Simulation numérique
    - Exemple LMJ
  - Imagerie virtuelle

Gestion des Processus

- Création/terminaison
- Contrôle/information/communication

Création : fork

- Effet
  - duplication du code
  - duplication des données
- Résultat
  - retourne 0 dans le fils
  - retourne le *pid* du fils dans le père

Création : exec

- Effet
  - Remplacement du code et des données du processus courant par celui du programme indiqué
- Variantes
  - execlp, execl, execle, execvp, execv, execve

Terminaison : wait

- Effet
  - Attend la fin d'un processus fils
    - void wait(int *status)
- Variante
  - Attend la fin d'un fils particulier
    - pid_t waitpid (pid_t pid, int *status, int options)

Communication

- Communication élémentaire
  - Fichiers partagés
  - Verrous
  - Signaux
- Communication plus complexe
  - Pipes nommés
  - Pipes

Signaux

- Ce mécanisme permet de *signaler* à un processus la survenue d'un événement particulier
- Envoie du signal : kill
- Réception d'un signal
  - Déclaration d'un *handler*
  - Déroutement du programme

Réception d'un signal

- Déclaration d'un *handler*
  - Définition
    - void handler(int signum);
  - Déclaration proprement dite
    - int signal(int signum, void (*handler) (int)))

Réception d'un signal

```
…
void  reinitialiser(int signum){
        …
}
…
Int main(void){
        …
        signal(SIGUSR1, reinitialiser);
        …
}
```

Envoie d'un signal

```
…
Int main(void){
        …
        kill(pid, SIGUSR1);
        …
}
```


Exercice

- Écrire un programme qui résiste au Ctrl-C, et qui affiche le message suivant quand il reçoit CTRL-C (i.e. SIGINT) :

  Merci d'éviter ^C pendant mon exécution ;-)

Pipes

- Un pipe est une voie de communication dans laquelle un processus peut écrire et dans laquelle un autre processus peut lire les données écrites par le premier.

Création d'un pipe

- int pipe(int fd[2])

```
…
Int fd[2];
…
pipe(fd);
If (fork()!=0){
        /* fils lecteur */
        close(fd[1]);
        …
}
If (fork()!=0){
        /* fils ecrivain */
        close(fd[0]);
        …
}
…
```

Redirection de l'entrée standard

- int dup(int fd)

```
…
        /* fils lecteur */
        close(fd[1]);
        close(0);
        dup(fd[0]);
        close(fd[0]);
…
```


Redirection de la sortie standard

- int dup(int fd)

```
…
        /* fils ecrivain */
        close(fd[0]);
        close(1);
        dup(fd[1]);
        close(fd[1]);
…
```

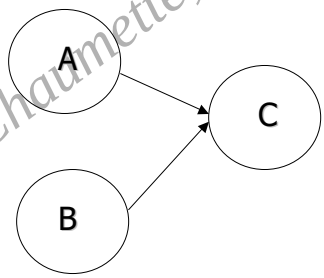
exec et les descripteurs de fichiers

- exec remplace la partie données.
- Les descripteurs de fichiers ouverts le restent (sauf ceux dont le drapeau *close-on-exec* est positionné).

Exercice

- Écrire un programme untee qui réalise l'architecture de processus suivante si on tape untee A B C.

15 minutes



Nouvelles problématiques

- Systèmes distribués
  - Création à distance
  - Communication à distance
- Hétérogénéité
  - Équilibrage de charge dynamique
- Mobilité de code
  - Migration
  - Accès