UNIVERSITÉ TOULOUSE III PAUL SABATIER Université de Toulouse

Licence - Mention Informatique

Programmation multitâche

Session 2 – 22 juin 2016
Durée : 1 heure 15 minutes

Aucun document autorisé
ATTENTION: on veillera à la présentation et aux commentaires

Une bibliothèque universitaire désire connaître le nombre d'étudiants qui entrent dans la bibliothèque, à tout instant, et l'afficher en permanence. On suppose que plusieurs entrées existent pour entrer dans la bibliothèque.
Un processus *Entree*, représentant une entrée, comptabilise les différents passages locaux, pendant un certain temps, puis fait connaître cette valeur à un processus *Compteur*. Ce dernier rafraîchit alors son affichage en conséquence. Cet affichage consiste à faire connaître le nombre total d'étudiants entrés dans la bibliothèque et le numéro de l'entrée par laquelle le plus grand nombre d'étudiants sont passés.

On veut implanter une solution en utilisant une communication/synchronisation par signaux et tubes.

Remarque : Afin de simuler le passage d'un étudiant par une porte et l'attente de cet événement, on pourra utiliser une fonction *detecterPassage()* qui consiste à se bloquer en attente de la lecture clavier d'un caractère (cette fonction est supposée **exister**).

Étude préliminaire

- Expliciter, par **schéma clair et bien commenté**, la communication entre les différents processus ainsi que les éventuels signaux utilisés.

Programmation

- Écrire le code du processus **Compteur**, celui du processus **Entree** ainsi que le **programme principal** permettant de les faire s'exécuter. Ce programme sera **paramétré** par le *nombre d'entrées* de la bibliothèque et le *délai de rafraîchissement* de l'affichage.

Barème (provisoire)

- Étude préliminaire : 6 points
- Programmation : 14 points

Quelques définitions UNIX

```
/* On suppose que les #include seront faits de la bonne manière,
➔ inutile de les préciser dans votre code */

int main(int argc, char **argv, char **env);

pid_t getpid (void);
pid_t getppid (void);
uid_t getuid (void);
uid_t geteuid (void);

pid_t fork (void);
void exit (int compteRendu);
void _exit (int CompteRendu);
pid_t wait (int *circonstances);
pid_t waitpid (pid_t pidAttendu, int *circonstances, int options);
/* options: WIFEXITED, WIFSIGNALED, WEXITSTATUS, WTERMSIG */

int execl (char *cheminAcces,
           char *arg0, char *arg1, ..., char *argN, NULL);
int execv (char *cheminAcces, char *tabArg[]);
int execlp (char *cheminAcces,
           char *arg0, char *arg1, ..., char *argN, NULL);
int execvp (char *cheminAcces, char *tabArg[]);
int execle (char *cheminAcces,
           char *arg0, char *arg1, ..., char *argN, NULL, char *env[]);
int execve (char *cheminAcces, char *tabArg[], char *env[]);

int creat (char *cheminAcces, mode_t droits);
int open (char *cheminAcces, int mode, mode_t droits);
/* mode : O_RDONLY, O_WRONLY, O_RDWR, O_APPEND, O_CREAT, O_EXCL, O_TRUNC */
int write (int numeroInterne, char *adresse, unsigned nombreTransmis);
int read (int numeroInterne, char *adresse, unsigned nombreDemande);
int close(int numeroInterne);
int dup (int numeroInterne);
int dup2 (int origine, int destination);
int pipe(int tube[2]);
```

```
int sigemptyset (sigset_t *ensSignaux);
int sigaddset (sigset_t *ensSignaux, int unSignal);
int sigdelset (sigset_t *ensSignaux, int unSignal);
int sigfillset (sigset_t *ensSignaux);
int sigismember (sigset_t *ensSignaux, int unSignal);

int sigprocmask (int actionSouhaitee,
                 sigset_t *ensSignaux, sigset_t *ancienEnsSignaux);
/* actionSouhaitee: SIG_SETMASK, SIG_BLOCK, SIG_UNBLOCK */

int sigpending (sigset_t *ensSignaux);

typedef void (*traitement) (int leSignal);

struct sigaction /* prédéfinie */ {
     traitement sa_handler; /* SIG_IGN, SIG_DFL, fonction */
     sigset_t   sa_mask;
     int        sa_flags;
};
traitement signal(int sigIntercepte, traitement monTraitement);

int pause();
int sigaction (int sigIntercepte,
     struct sigaction *nouvelleAction, struct sigaction *ancienneAction);
int sigsuspend (sigset_t *ensSignaux);

int kill(int pidDestinataire, int sigEmis);
unsigned alarm(unsigned duree);
```