Rapport Exo6 TP0

Belmouloud Mustapha Abdellah
groupe 1

l’enoncé demande de creer 3 processus enfants un pour conpter les secondes, l’autre pour les minutes et l’autre pour les heures, donc on a besoin de: trouver un moyen de communication entre les processus enfant, qui est le probleme principal de l’exercice

Solution de l’exo:
on vat utiliser sigqueue et sigaction pour envoyer des signaux SIGUSR1 avec data sure le envoyeur et des variable, il dependent sure des structure predefini dans c

Syntaxe:

```
#define _POSIX_C_SOURCE 200809L
#include <signal.h>
#include <unistd.h>
---------sending---------
union sigval sv;
sv.sival_int = value;
sigqueue(pid, SIGUSR1, sv)

--------receiving---------
void handler(int sig, siginfo_t *info, void *context) {
printf("Got value: %d\n", info->si_value.sival_int);
}
int main(void) {
struct sigaction sa = {0};
sa.sa_sigaction = handler; //nom de fonction a declacher
sa.sa_flags = SA_SIGINFO;

sigaction(SIGUSR1, &sa, NULL);
```

D’abord on doit envoyer le PID des processuses H et M a les processuses M et S, pour pouvoir declancher l’incrementation, apres, en vas utiliser une boucle qui incremente chaque seconde et envoi le resulta aux pere pour afficher, si les second == 60 doc en

vas envoyer un signal a le processus M pour incrementer les minutes et envoyer a le pere, on doit egalement utiliser pause(); attendre l'ariver des signaux dans la boucle et une variable ready, pour assurer que les processus recoient les pid d'aure processus avans d'envoyer :

le processus S:

```
S = fork();
if(S==0){
//receive M id
struct sigaction sa = {0};
sa.sa_sigaction = Srec;
sa.sa_flags = SA_SIGINFO;
sigaction(SIGUSR1, &sa, NULL);

while(!ready){pause();}

union sigval svv;

while(true){

sleep(1);

seconds++;
svv.sival_int = seconds;
sigqueue(F, SIGUSR1, svv);

if(seconds == 60)
{
seconds=0;
union sigval sv;
sv.sival_int = 0;
sigqueue(M, SIGUSR1, sv);
}

}

exit(0);
}
```

le processus M:

```
M = fork();
if(M==0){
//recieve H id and S signals
```

```
struct sigaction sa = {0};
sa.sa_sigaction = Mrec;
sa.sa_flags = SA_SIGINFO;
sigaction(SIGUSR1, &sa, NULL);

while(!ready){pause();}

minutes = 0;
union sigval svv;

while (true){
pause();

minutes++;
svv.sival_int = minutes;
sigqueue(F, SIGUSR1, svv);

if(minutes == 60)
{
seconds=0;
union sigval sv;
sv.sival_int = 0;
sigqueue(H, SIGUSR1, sv);
}
}

exit(0);
}
```

le processus H

```
H = fork();
if(H==0){
//receive M signals
struct sigaction sa = {0};
sa.sa_sigaction = Hrec;
sa.sa_flags = SA_SIGINFO;
sigaction(SIGUSR1, &sa, NULL);


while(true){
pause();
houres++;
union sigval svv;
svv.sival_int = houres;
sigqueue(F, SIGUSR1, svv);
}

exit(0);
}
```

pour un bon resultat il faut mofifier la function de M qui recois les signauc pour distinguer entre les signaux de pere et les singaux de S:

```
void Mrec(int sig, siginfo_t *info, void *context){
if(info->si_pid == F){
H = info->si_value.sival_int;
ready = true;
}
if(info->si_pid == S){
//
}
}
```

et finalement la function de pere qui affiche les resultats, on doit utiliser fflush() pour assurer l’affichage immediat des resultats et verifier si les secondes ou les minute egale a 60 pour ne pas afficher deux fois, aussi il faut prendre en compte que le recois des minutes n’est declanché sauf si les second sont 0 car si second==60 donc on doit revenir a 0, et la meme chose pour les heures et le minutes:

```
void Frec(int sig, siginfo_t *info, void *context){
if(info->si_pid == S){
seconds = info->si_value.sival_int;
if(seconds != 60){
printf("%02d : %02d : %02d \n", houres, minutes, seconds);
fflush(stdout);
}
}
if(info->si_pid == M){
minutes = info->si_value.sival_int;
if(minutes != 60){
printf("%02d : %02d : %02d \n", houres, minutes, 0);
fflush(stdout);
}
}
if(info->si_pid == H){
houres = info->si_value.sival_int;
printf("%02d : %02d : %02d \n", houres, 0, 0);
fflush(stdout);
}
}
```

Conclusion:
les method traditionelle kill et signal ne sont pas suffisantes pour atteindre notre bute, donc il faut utiliser sogqueue et sigaction ou des pipeline, la syntaxe est basé sure des structure predifini qui nous permet a envoyer des variables et des informations sur les processus envoyant comme leur pid, comme ça on peut cree sort de hyarchy pour les processus et declacher l'un avec l'autre, sigquee aussi resolue le probleme de perte des signaux car ils sont mets en fille d'attente