Compte rendu Info0403

Daunique Wilfried

Gigout Thomas

Le projet:

Créer un système d'élection de processus au travers de l'architecture de la figure

Les contraintes :

-Choisir l'identifiant du processus aléatoirement entre 1 à n (n étant le nombre de processus.

-retransmettre les messages en incrémentant une valeur hope.

-évaluer si le processus est battus ou élu suivant son id.

-sauvegarde une trace dans un fichier créer

-retransmettre les messages reçu en incrémentant le hope.

Solutions envisagés :

Tout d'abord il nous fallait créer une boucle de processus comme sur la figure 1. Nous sommes donc aller voir dans nos cours car nous nous souvenions avoir fais un exercice sur un schéma de n processus discutant via n tube. On a alors alloué un pointeur de int avec 2*n case. On a ensuite pipe ce tableau pour pouvoir s'en servir de tube.

```
int *tube = (int*)malloc(2*n*sizeof(int));

for(int i=0 ; i<n ; i++){
  pipe(&tube[2*i]);
}
```

Il fallait alors gérer la création des processus chacun d'eux devait avoir un nombre aléatoire comme id pour pouvoir procéder à une élection.

Cependant comme les processus se crée trop rapidement pour qu'un srand() soit effectif on met alors celui-ci à la puissance i puis n (le i est initialiser dans la boucle for créant n processus . On rentre alors la valeur dans un tableau de même taille que le tube pour pouvoir enregistrer l'id des processus et le valeur du hope (en effet quand le message est reçu on l'incrémente de 1 et quand il repart on le réincrémente de 1) .Cette étape a pris pas mal de temps car

n’ayant pas crée une struct nous emmagasinons trop de données dites inutiles et le tableau étant trop grand il fallait a chaque fois voir si i était pair ou impair pour remplir le tableau avec en impair la valeur id des processus et en pair la valeur hop.

```
for(int i=0 ; i<n*2 ; i++){

  srand(time(NULL)*i^n);
  c = rand()%n+1;
  tab[i*2]=c;
  if(i%2==1){
    if(i==1){
      tab[1]=0;
      // printf("test");
    }
    else{
      // printf("test");
      tab[i] = tab[i-2]+1;
    }
```

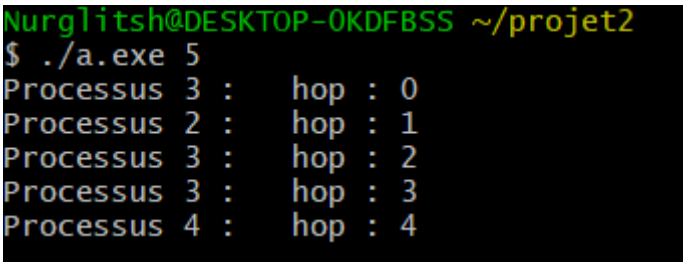
Ensuite viens la partie qui nous a retenu le plus de temps , nous avons reprit l’exercice sur les n tubes et avons récréer les même boucle cependant la boucle else ne nous permet plus de faire d’affichage et semble faire continuer le processus à l’infini.

```
    if(fork()==0){
      //  waitpid(p,&status,WUNTRACED|WCONTINUED);
      in = tube[2*i];
      out = tube[2*i+3];
      // printf("%d\n",i);
      // printf("%d\n",c);
      read(in, &c, sizeof(int));
      write(out, &c, sizeof(int));
      exit(0);
    }
    else{
      in = tube[2*n-2];
      out = tube[i];
      write(out, &c, sizeof(int));
      read(in,&c,sizeof(int));
     }
    exit(0);
  }
}
```

Dans le if nous évaluons le cas où le processus fils reçois l'information dans le tube et écrit au père (ex : i=0 on reçoit de la case 0 et on écrit dans la case 3 du pipe)

On ne passe que la valeur de l'id qui servira a savoir si le processus est battus ou non (test si t[hope*2-1] > au message reçu)

Voici l'affichage lors de la désactivation de la boucle else qui nous créait une boucle infini.



Donc on crée 5 processus et le hop passe a 4 car le message a été passer 5 fois.

On a donc penser que c'était due à un problème d'exit(0) . On a donc essayé d'en enlever puis dans rajouter mais rien ne semblait résoudre mon problème.

Il nous fallait maintenant repartir de 0 ne trouvant plus aucune solution pour résoudre ce problème.

Cependant le temps venait à manquer et nous nous sommes donc dit que cela ne valait plus la peine de recommencer à 0 et on est donc parti du principe que cela fonctionnait et nous nous sommes alors dédiés à la création de l'élection.

L'élection paraissait infaisable au vu de notre code mais comme expliquer ci-dessus il nous fallait comparer la valeur

passer dans le tube précédant à la valeur actuelle si celle-ci se trouve être supérieur à la valeur actuelle on continue de le passer sinon on passe la nouvelle valeur.

On a alors décider d'utiliser le tableau que nous avons créer pour enregistrer les hop et les messages

```
*tab = (int *) malloc(2 * n * sizeof(int));
```

```
if (fork() == 0) {
  //  waitpid(p,&status,WUNTRACED|WCONTINUED);
  in = tube[2 * i];
  out = tube[2 * i + 3];
   //printf("%d\n",i);
  printf("%d\n",c);
  if (tab[i*2-2]>tab[i*2]) {
    read(tube[2*i-2], &c, sizeof(int));
    write(tube[2*i-2], &c, sizeof(int));

  } else {
    read(in, &c, sizeof(int));
    write(out, &c, sizeof(int));
  }

  exit(0);
```

L'affichage nous donne seulement la dernière valeur entrer dans le message sans rentrer dans nos conditions.

Si cela fonctionnait on aurait pu se reconcentrer sur nos processus mais avançant avec ce fardeau on ne pouvait pas vraiment savoir où les erreurs se trouvaient.

Exemple d'affichage :

```
$ g++ projet.cpp -o a.exe

Nurglitsh@DESKTOP-OKDFBSS ~/projet2
$ ./a.exe 5
 Processus : 0   ID : 3
 Processus : 1   ID : 2
2
 Processus : 2   ID : 4
 Processus : 3   ID : 3
3


Nurglitsh@DESKTOP-OKDFBSS ~/projet2
$ ./a.exe 5
 Processus : 0   ID : 3
 Processus : 1   ID : 4
4
 Processus : 2   ID : 4
 Processus : 3   ID : 1
1
```

Il nous affiche maintenant plus que 4 processus sur les 5 que l'on arrivait à afficher auparavant on à alors retester en désactivant la boucle else.

On a alors compris pourquoi l'exécutable ne fonctionnaient pas à cause de notre boule n*2 au début du programme celui- ci crée 2*n programme n étant la valeur rentrer.

On a alors enlever le n*2 et on a pensé que nous pourrions faire une struct plus tard pour rentrer les valeur dans tab notre tableau qui devait garder les valeur des messages

Après quelque modification notre processus crée finalement le nombre de processus attendu comme au début.

Lors du premier affichage suite à cette modification quelque chose nous interpella.

```
Nurglitsh@DESKTOP-OKDFBSS ~/projet2
$ ./a.exe 7
 Processus : 0   ID : 2
 Processus : 1   ID : 4
 Processus : 2   ID : 1
 Processus : 3   ID : 7
4
 Processus : 4   ID : 7
 Processus : 5   ID : 6
7
 Processus : 6   ID : 3
6
```

Il semblait que l'élection n'avait pas le temps de ce faire donc il fallait faire attendre les fils au père.

Cependant après l'ajout de cette ligne rien ne change.

Il nous restait alors peux de temps nous nous sommes alors dit comme précédemment qu'on devrait au moins essayer d'afficher ses résultats dans un fichier.

On ajoute alors les lignes suivantes :

```
if(f=open(nom,O_CREAT|O_WRONLY,777)){
  for (int i=0 ;i<n;i++){
      write(f,(void*)&tab[i],sizeof(int));
   }
   printf("\n");
}
close(f);
```

Arrivant à la limite du temps pour pouvoir rendre le projet on a alors décider de faire ce compte rendu et d'essayer d'imaginer ce que l'on pourrait ajouter au programme et ce que nous aurions pu faire pour le faire fonctionner.

Ce que l'on aurait aimé faire :

C'est un peux déçu que nous rendons ce projet , si l'on avait eu plus de temps on aurait pu faire fonctionner l'élection qui semble bancale, réfléchir à une condition plus concrète. On aurait eu le temps aussi de tout recommencer et de faire une struct à la place de nos deux valeurs que l'on rentrait à chaque passage. Cela nous aurait permis de moins nous creuser la tête au sujet de ce programme.

Ensuite on aurait pu se creuser la tête pour faire fonctionner l'écriture dans le fichier qui ne semblait pas marcher alors que l'on a répété les mêmes étapes que lors du tp2 ex3.

On aurait ensuite pu gérer les cas où 2 processus avait le même ID en choisissant si on voulait en sleep 1 ou non en gérant les signaux

Conclusion :

Je pense que ce n'est pas vraiment par manque de temps, mais par manque d'efficacité et à cause des erreurs qui se succédaient qui on fait que nous n'avons pas pu finir le programme à temps et qu'il est incomplet.

Nous avons tout de mêmes essayer de faire le maximum de chose malgré les erreurs en essayant au maximum que la partie globale du programme marche (création et ajout des valeurs aléatoires).