UNIVERSITE DE CAEN
Licence d’Informatique

DEVOIR SYSTEME

Julien VAN DEN BOSSCHE / Benoît MOULIN
jbossche@etu.info.unicaen.fr / bmoulin@etu.info.unicaen.fr
Groupe 3.2

Le programme à réalisé permet de décomposer un nombre en facteurs premiers par l'intermédiaire de plusieurs processus ouvriers. Un processus maître permettra le contrôle de ces ouvriers. On verra donc dans ce programme la communication entre les processus, la manière d'envoyer, de recevoir les informations échangées. On expliquera aussi comment le maître peut agir sur ses ouvriers pour leurs faire suspendre ou continuer leurs tâches.

A.Exécutable init

Cet exécutable prend en paramètre le nombres d'ouvriers. On vérifie d'abord la validité du nombre rentré car il doit être compris entre un et dix.
On rentre dans une boucle qui crée le nombre d'ouvriers voulu ainsi qu'un tube ordinaire pour chacun d'entre eux.
Pour chaque processus créé, lorsque que l'on rentre dans sa section (processus fils), on ferme l'entrée en écriture du tube ordinaire(l'ouvrier étant lecteur dans le tube ordinaire). On redirige l'entrée standard du fils vers l'entrée en lecture du tube ordinaire (par dup2) puis on ferme cette entrée. Ensuite on recouvre ce processus par l'exécutable « ouvrier ».
Quand on rentre dans la section père, on ferme l'entrée du tube en lecture et on redirige le descripteur i+3 vers l'entrée du tube en écriture. On ferme l'entrée en écriture du tube.
Quand on sort de la boucle on recouvre le processus père par l'exécutable « maître » en passant le nombre d'ouvriers en paramètre.

B.Exécutable maître

Cet exécutable prend en argument le nombre d'ouvriers.
Il contient plusieurs attributs. On a un tableau d'entiers qui contient les 100 premiers nombres premiers. Il contient trois *struct* qui vont permettre d'envoyer et de récupérer les données avec les processus ouvriers. On a choisit de communiquer en utilisant des *struct* pour éviter l'opération de conversion en *char** du message à échanger (Ex : si on communique l'entier 3, on est obligé de convertir cet *int* en *char** si on utilise pas de *struct*).
On a aussi un tableau à deux dimensions qui contiendra les *pid* et l'état des processus ouvriers (tâche terminée ou non).

Première étape : envoi et réception de messages pour l'initialisation des données

- On crée un tube nommé « canal » qui permettra de communiquer avec les fils. Le processus maître sera en mode lecture.

- Envoi des nombres premiers aux ouvriers par les tubes ordinaires : le message envoyé est un *struct* possédant comme attributs un booléen « fin » et un entier « nombre ». On calcule le quotient de la division euclidienne de la taille de la table des nombres premiers par le nombre d'ouvriers. Appelons p ce nombre. Les n-1 ouvriers recevront p nombres premiers et le dernier ouvrier recevra le reste de la division euclidienne. Pour le dernier nombre premier envoyé pour chaque processus on initialise son booléen « *fin* » à *true* ce qui permettra de connaître la fin de lecture dans les processus ouvriers.

- Lecture des *pid* des ouvriers via le tube « canal » : on ouvre le tube « canal » en mode lecture et tant que le compteur est inférieur aux nombres d'ouvriers on lit dans ce tube un *struct* et on insère dans la table des *pid* l'attribut *pid* de ce dernier. On initialise l'état de l'ouvrier à 0 dans la table.

Deuxième étape : boucle principale, attente de réception de facteurs premiers

Dans cette boucle on se met en attente de la réception d'un facteur dans le tube nommé « canal ». On sortira de la boucle quand le quotient (divisions successives du nombre par les facteurs reçus est égal à 1). Cela signifie que l'on possède tous les facteurs. On regarde aussi si l'ouvrier qui vient d'envoyer son message a fini sa tâche (si le nombre envoyé est un). Ensuite on affiche les *pid* des ouvriers et leurs états. Si un ouvrier n'a pas fini sa tâche on lui envoi un signal qu'il captera pour stopper son exécution et se mettre en attente d'un nouvel entier à décomposer (*sigsuspend*).
On redemande à l'utilisateur un nouveau nombre ou il peut sortir du programme.
S'il sort on ferme les tubes, on tue les processus ouvriers.

C.Exécutable ouvrier

Dans cet exécutable le processus va communiquer avec les autres processus (maitre) via des messages envoyés dans les tubes (ordinaires et nommés). Ces messages sont des *struct*.
Il y a deux types de *struct* :

- message_1 qui a comme attributs un nombre (*int*) et un booléen (nommé *fin*).
- message_2 qui a comme attributs deux *int* : facteur et *pid* .

Première étape : envoi et réception de messages pour l'initialisation des données

La lecture dans un tube étant destructrice il faut respecter l'ordre dans lequel on a envoyé les messages dans le processus maître.

- Réception des nombres premiers envoyé par maître :

La structure à lire dans le tube a comme attributs un entier et un booléen. On lit dans le tube tant que le booléen n'est pas à *true* et on insère le nombre du *struct* dans une liste. Si sa valeur est *true* on arrête donc la lecture car le dernier *struct* envoyé par la maître a mis son booléen à *true*.

- Envoi du *pid* au processus maître

On ouvre le tube « canal » en mode écriture (open). On envoie (*write*) le *pid* dans le tube via le *struct* qui a pour attributs deux entiers (*facteur* et *pid*). Ici on ne se sert pas de facteur.

- Réception du nombre à décomposer :

On réceptionne le nombre à décomposer via le tube ordinaire. On lit dans ce tube un *struct* et on stocke l'attribut nombre du *struct* dans *int* nombre.

Deuxième étape : boucle principale, recherche de facteurs premiers

Dans cette deuxième partie du programme on rentre dans une boucle infinie tant que la liste des nombres premiers n'est pas explorée.
On parcours la liste des nombres premiers (par un itérateur). Si le premier nombre de la liste est un diviseur du nombre à décomposer alors on envoie ce nombre au processus maître via le tube « canal » (dans un *struct* avec le *pid* et le nombre). Le nombre à décomposer, pour ce processus, devient alors le résultat de la division du nombre à décomposer par le facteur trouvé. On garde l'itérateur à la même place.
Si le nombre n'est pas un diviseur alors on avance d'un cran dans la liste et on recommence.
Une fois la boucle terminée on prévient le maître que l'ouvrier à fini. On envoie donc un *struct* au maître via le tube « canal ». Ce *struct* à son attribut facteur à zéro et son *pid* égal au *pid* du processus.

Réception des signaux

Au préalable, nous avons pris soin d'installer un handler de réveil (par *signal(SIGUSR1, …)*). Une fois la décomposition trouvée au niveau du processus maître, ce dernier envoi un signal (*SIGUSR1*) aux fils n'ayant pas terminés leurs tâches. Dans la boucle principale de décomposition nous avons installé un masque de signal (*sigprocmask(…)*).Les processus ouvriers sont mis en attente d'un nouveau nombre à décomposer.