HE VINCI 5 juin 2023

BINV2181 : Linux : Programmation distribuée

J. Vander Meulen, A. Legrand, O. Choquet, X. Gillard

Durée de l'examen : 120 minutes (pas de sortie durant les 30 premières minutes)

Consignes importantes :

- Nous vous demandons de faire particulièrement attention à ce que votre code ne produise par d'erreurs de compilation. Notez que nous vous demandons de compilez votre code avec les flags utilisés lors du cours :

```
gcc -std=c11 -pedantic -Wall -Wvla -Werror
    -Wno-unused-variable -D_DEFAULT_SOURCE
```

- A la fin de l'examen, soumettez vos fichiers sur EvalMoodle. Voir la page 5 de cet énoncé pour plus d'informations.

1. Pipes (10/20 points)

Ecrivez un programme « pipes » qui ne reçoit aucun argument sur la ligne de commande et qui crée deux processus (père et fils) qui communiquent via deux pipes.

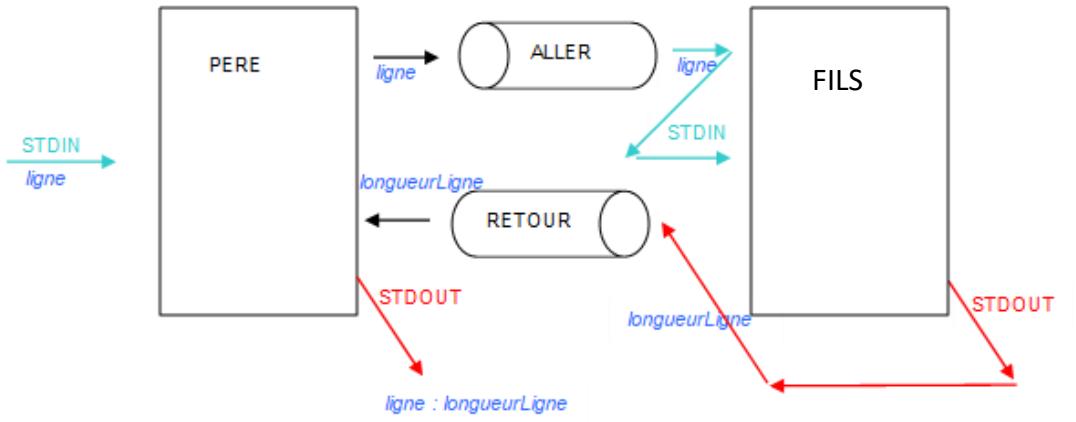
Le père lit des lignes à l'entrée standard et les envoie à son fils via le premier pipe.

Le processus fils redirige son entrée standard sur la lecture du premier pipe et sa sortie standard sur l'écriture du deuxième pipe puis lit les lignes envoyées par son père, calcule leur taille et écrit celle-ci à la sortie standard.

Le père affiche la ligne envoyée à son fils suivie de sa taille.

Quand l'utilisateur introduit un ctrl-D à la place d'une ligne à l'entrée standard, le père ferme l'écriture sur le premier pipe afin d'envoyer une fin de fichier à son fils qui se termine en affichant le nombre de lignes qu'il a traitées avant de fermer l'écriture sur le deuxième pipe pour avertir son père qu'il peut lui aussi se terminer après avoir écrit la dernière information reçue (à savoir le nombre total de lignes traitées).

Voici un schéma représentant le programme « pipes » :



Voici le résultat attendu :

```
olivier@bilout:~/Téléchargements/5eme/pipes_solution$ ./pipes
bonjour
bonjour : 7
az
az : 2
Le programme a traité 2 lignes
olivier@bilout:~/Téléchargements/5eme/pipes_solution$
```

Indications utiles :

1. Vous avez un répertoire pipes dans les ressources avec le Makefile et la librairie utils_v2. Il ne reste qu'à écrire le fichier pipes.c
2. Pour votre facilité, utilisez la fonction printf pour les affichages à l'écran c'est-à-dire dans le père.
3. Les lignes lues au clavier et envoyées par le père à son fils ne feront jamais plus de 80 caractères.
4. Aucun besoin de signaux pour fermer proprement les 2 programmes.

Pour cette question, il vous est demandé de :

1. compléter le programme *pipes.c*

2. IPC (10/20 points)

Cette question est relative au « tri par comptage »[1]. Pour rappel, c'est un algorithme de tri qui utilise une table de fréquences.

Notre construction de cet algorithme fonctionne de la manière suivante. Nous avons un processus père qui prend en entrée un tableau de n éléments (e.g. le tableau suivant de 6 éléments {8, 2, 10, 8, 7, 2}) compris entre 0 et 999.

1. Le processus père initialise correctement **un tableau** de 1000 int **en mémoire partagée**, ainsi qu'**un sémaphore**.
2. Le processus père crée **un pipe**.
3. Le **père crée un certain nombre de fils Ce nombre est passé en paramètre.**
4. Le père **écrit toutes les valeurs** du tableau d'entrée dans le pipe.
5. Chaque fils doit faire les deux actions suivantes, dans le bon ordre :
    a. Configurer correctement le pipe ; il doit donc **fermer correctement les deux côtés du pipe**.
    b. **Exécuter une boucle qui lit sur le pipe des valeurs x**. Pour chaque valeur x, il **incrémente la case à l'indice x** du tableau partagé.
6. Pendant ce temps, le **père attend la fin de tous ses fils**.
7. Quand tous ses fils sont terminés, le père **affiche le tableau trié**.
8. Pour finir, il **supprime** la mémoire partagée et le sémaphore.

Le programme **« countingsort.c »** prend en argument n+2 nombres (e.g. `./countingsort 3 6 8 2 10 8 7 2`). Le premier argument est le nombre de fils (dans notre exemple, il y a 3 fils). Le deuxième argument est égal à n. Il permet d'indiquer la taille du tableau. Dans notre exemple : 6. Ensuite, il prend n nombres représentant les n valeurs du tableau. Dans notre exemple : 8, 2, 10, 8, 7 et 2.

Voici un exemple d'exécution du programme :

```
$ ./countingsort 3 6 8 2 10 8 7 2
2 2 7 8 8 10
```

[1] https://fr.wikipedia.org/wiki/Tri_comptage

Pour cette question, il vous est demandé de :

2. compléter le programme *countingsort.c*
3. compléter le fichier *Makefile*.

3. Fichiers fournis

Sur EvalMooldle, vous trouverez les fichiers suivants :

1. pipes.c (Q1)
2. countingsort.c (Q2)
3. Makefile (Q1 & Q2)
4. utils_v2.h (Q1 & Q2)
5. utils_v2.c (Q1 & Q2)
6. HEVINCI-2022-2023-BINV2180-EXAMEN-JUIN.pdf (ce fichier)
7. Moodle.zip

4. À remettre

Sur EvalMoodle, les fichiers pipes.c, Makefile et countingsort.c complétés.