ANNÉE 2004-2005 SESSION DE MAI 2005

**Étape :** Licence Sciences Technologies (semestre 4) **UE :** INF204
**Épreuve de :** Programmation Système **Durée :** 1h30
**Date :** 17 mai 2005
**Heure :** 11h00
Documents notes de cours autorisées
Épreuve de Monsieur Namyst

***Lisez l'intégralité du sujet*** *avant de commencer à répondre aux questions. Certaines peuvent être traitées même si vous n'avez pas répondu aux précédentes.*

# 1 Processus légers

Décrivez brièvement les principales différences entre un processus et un thread. Citer des exemples d'utilisation avantageuse des threads par rapport aux processus.

# 2 Signaux

On souhaite, dans un programme sous Linux, déterminer l'adresse du point de rupture du tas (c'est-à-dire la première adresse mémoire invalide en parcourant le tas vers le haut) ainsi que le point de rupture de la pile (c'est-à-dire la première adresse mémoire invalide en la parcourant vers le bas).

On se propose pour cela d'utiliser la propriété qu'un accès à une adresse invalide provoque l'envoi d'un signal `SIGSEGV` au processus. L'idée est la suivante : pour trouver le point de rupture du tas, il suffit, à partir d'une adresse valide du tas, de le parcourir par adresses croissantes en effectuant un accès en lecture à chaque fois. Lorsque l'accès en lecture sera au-delà du point de rupture, il suffira de rattraper le signal `SIGSEGV`, de sortir proprement du traitant de signal et de la boucle de parcours pour passer à la suite du programme, en l'occurence la recherche du point de rupture de la pile.

Voici le squelette du programme `rupture.c` :

```
void *adresse_valide_tas(void)
{ ... }

void *adresse_valide_pile(void)
{ ... }

int main()
{
  char *rupture_tas = NULL, *rupture_pile = NULL;
  ...
  printf("Zone interdite : entre %p et %p\n", rupture_tas, rupture_pile);
}
```

**Question 1** Indiquez une méthode pour trouver une adresse valide quelque part dans la pile (peu importe laquelle, il suffit d'en connaître une pour permettre de débuter le parcours). Donnez le code de la fonction correspondante `adresse_valide_pile` qui calcule et renvoie une telle valeur.

**Question 2** Même question concernant le tas : indiquez une méthode pour y trouver une adresse valide et donnez le code de la fonction `adresse_valide_tas`.

**Question 3** Il s'agit maintenant de mettre en place le traitant de signal pour `SIGSEGV`, ainsi que le code qui permettra de revenir dans la fonction `main` une fois le signal délivré. Complétez le code du programme `rupture` de manière à récupérer dans la variable `rupture_pile`, avant l'appel à `printf`, l'adresse du point de rupture de la pile. N'oubliez pas de donner le code de la fonction "traitant de signal". Vous pouvez bien sûr ajouter des variables globales si nécessaire.

**Question 4** Complétez le programme de manière à récupérer également l'adresse du point de rupture du tas dans la variable `rupture_tas`, avant l'appel à `printf`. Est-il nécessaire d'utiliser un second traitant de signal ? Expliquez.

# 3 Sortie standard dédoublée

On souhaite disposer d'une commande `dedoubler` grâce à laquelle on pourrait exécuter un programme (quelconque) en récupérant automatiquement une copie de sa sortie standard dans un fichier. En quelque sorte, il s'agit de dédoubler chacun des caractères envoyés sur sa sortie standard : un exemplaire est reproduit sur la sortie standard par défaut alors que l'autre est enregistré dans un fichier.

Le programme `dedoubler` prend en premier argument le nom du fichier à utiliser pour la sortie. Les arguments suivants constituent la commande à exécuter. Pour fixer les idées, voici un exemple d'utilisation de ce programme :

```
> dedoubler fichier ls -la
drwxr-xr-x  2 rnamyst rnamyst    80 2005-04-28 22:29 ./
drwxrwxrwt  7 root    root      180 2005-04-28 22:27 ../
-rw-r--r--  1 rnamyst rnamyst     0 2005-04-28 22:29 fichier
-rw-r--r--  1 rnamyst rnamyst 20041 2005-04-28 22:27 toto.pdf
> cat fichier
drwxr-xr-x  2 rnamyst rnamyst    80 2005-04-28 22:29 ./
drwxrwxrwt  7 root    root      180 2005-04-28 22:27 ../
-rw-r--r--  1 rnamyst rnamyst     0 2005-04-28 22:29 fichier
-rw-r--r--  1 rnamyst rnamyst 20041 2005-04-28 22:27 toto.pdf
>
```

**Question 1** Pour réaliser le dédoublement de la sortie standard d'un processus, une solution est de rediriger sa sortie standard vers un *tube* à l'extrémité duquel un second processus pourra alors facilement extraire tous les caractères et les sauver dans un fichier en même temps qu'il les affichera sur sa sortie standard.

Le programme `dedoubler` va donc créer un processus fils et un tube partagé par le père et le fils. Le fils, par exemple, pourra exécuter la commande spécifiée en paramètre, tandis que le père se chargera de l'affichage et de l'écriture dans le fichier.

Donnez le code complet du programme `dedoubler`. Pour exécuter la commande, on pourra utiliser l'appel système `execvp` qui possède le profil suivant (`file` désigne le nom de l'exécutable et `argv` le tableau des paramètres) :

```
int execvp(const char *file, char * argv[]);
```

**Question 2** Une solution plus simple serait d'utiliser un programme intermédiaire (que l'on appelera `tee`) dont la tâche serait simplement de lire son entrée standard et de la reproduire à la fois sur la sortie standard et dans un fichier dont le nom serait passé en paramètre. Voici un exemple d'utilisation de `tee` :

```
> echo Salut a tous | tee toto
Salut a tous
> cat toto
Salut a tous
>
```

Donnez le code du programme `tee`.

**Question 3** En utilisant `tee`, donnez une version plus simple du programme `dedoubler`.

**Question 4** En regardant attentivement la sortie donnée en exemple pour la commande `dedoubler`, on voit la ligne suivante apparaître :

```
-rw-r--r--  1 rnamyst rnamyst     0 2005-04-28 22:29 fichier
```

Expliquez pourquoi.

Avec votre implémentation de `dedoubler`, est-il possible que le fichier n'apparaisse pas toujours ? Est-il possible que la taille observée soit non nulle ? Pourquoi ?