Séance de TP N°1
`01.entrees-sorties` – Traitement des erreurs et entrées-sorties

Méthode de travail pour tous les TPs de cet enseignement

Création du répertoire de séance

Créez sous votre répertoire racine un répertoire "l3i6" qui vous servira pour toutes vos activités dans cet enseignement.
Pour chaque séance de TP, vous commencerez par créer un répertoire de séance "*~/l3i6/nn.libellé*", avec "*nn*" le numéro de la séance sur deux chiffres[1] et "*libellé*" un identificateur résumé mnémotechnique. L'identificateur du TP est aussi indiqué dans le titre du sujet et sera utilisé pour tous les documents fournis par les enseignants pour ce TP : fichiers de fourniture, corrigé et sujet sur le site Web, etc. Donc, pour aujourd'hui vous créez un répertoire de séance "*~/l3i6/01.entrees-sorties*" et vous vous placez sur ce répertoire.

Dépliage de l'archive de fourniture

Pour chaque séance, vous aurez à récupérer une archive en format "*.tgz*" que vous déplierez dans votre répertoire de séance. Cette archive se trouvera sur un répertoire de votre enseignant à l'endroit indiqué (ou sur le site des TPs). Elle contient des fichiers de programmes à compléter ou à modifier, un script **configure** et un Makefile générique "*Makefile.gen*". Il faudra toujours exécuter, avant toute chose, le script configure qui produit le fichier "Makefile" prêt à l'emploi, pour chaque TP, et en fonction de la machine UNIX utilisée : LINUX, CYGWIN, SOLARIS, OSF/1, MAC OS/X (Darwin). Ces trois derniers systèmes étant peu utilisés, le script "*configure*" et les bibliothèques associées ne sont plus maintenues. S'il y a des petits problèmes de portabilité des bibliothèques ou des fichiers d'en-tête, il faudra vous-même les adapter.
Si vous devez modifier le "*Makefile*", il faut toujours modifier le "*Makefile.gen*" (Makefile générique) et réexécuter le script configure.

Nettoyages de fin de séance

Pour éviter d'avoir le disque saturé, il faut prendre l'habitude de nettoyer son répertoire de TP en fin de séance. Cela s'obtient généralement en exécutant la commande **make veryclean**. De même, il vaut mieux commencer un semestre en effaçant tous les fichiers des années ou des semestres précédents qui sont complètement inutiles. En particulier, si vous avez un répertoire "*~/Library*", vous pouvez l'effacer sans scrupule. Les autres répertoires peuvent être utilement compressés après nettoyage. Il faut de temps en temps effacer vos répertoires de caches de navigation Internet, et les fichiers fournis par vos enseignants que vous pouvez toujours récupérer. Utilisez la commande **du ~/*** pour avoir une idée de votre occupation mémoire. Vous ne devriez pas occuper plus que 2 ou 3 Mo chacun. La commande **df ~** vous indique le taux d'occupation du disque qui vous est attribué.

[1] Pensez à mettre un 0 en tête, si nécessaire, pour que les listages alphabétiques fonctionnent correctement.

1. Recopie de fichier version 1

Recopiez dans votre répertoire de séance l'archive fournie "*~lahire/l3i6/01.entrees-sorties/Fourniture1.tgz*" et déballez-là.
Ecrivez une fonction *void copierFichier1(int source, int destin)* qui recopie dans un fichier "*destin*" le fichier "*source*". "*destin*" et "*source*" sont deux descripteurs de fichiers ouverts dans le programme appelant dans les modes appropriés pour la lecture ou l'écriture. Cette fonction ne doit pas faire d'hypothèse sur la nature des médias associés aux fichiers. Elle doit tester les comptes-rendus de tous les appels système et ne comporter que des appels système pour effectuer le traitement (pas les primitives de la bibliothèque du langage C). On utilisera un tampon de taille BUFFSIZE = 100 000 octets et à la sortie de la fonction, celle-ci affichera la taille du tampon utilisé, la taille du fichier lu et le nombre d'appels système pour la lecture du fichier source et l'écriture du fichier destination.
Dans cette version, le traitement des erreurs se fera en utilisant les primitives standard du système Unix, comme la consultation de la variable *errno* et les primitives associées. Elle n'utilisera pas (encore) les fonctions de Stevens ou les macros de Rousseau. Par contre, vous pourrez utiliser les primitives d'entres-sorties du langage C (*fprintf*...) pour écrire des messages sur la voie d'erreur.

Ecrire un programme de test **tcopier1** qui appelle cette fonction en ouvrant les fichiers source et destination à partir de noms fournis sur la ligne de commande, avec la syntaxe :*tcopier1 source destination*. Par convention, si source ou destination sont des nombres, ils indiqueront un numéro de voie d'E/S de fichiers déjà ouverts.

Faire les tests suivants :

```
~> tcopier1 tcopier1.c fich2
# On doit avoir un résultat du genre :
Taille tampon: 100000 ; Taille fichier lu: 3837 ; Nb read: 1 ; Nb write: 1
~> wc tcopier1.c
108 493 3837 tcopier1.c
~> diff tcopier1.c fich2
# Il ne doit pas il y avoir de différences
~> cp Fourniture1/MOTS fich1
~> tcopier1 fich1 fich2
# On doit avoir le résultat suivant
Taille tampon: 100000 ; Taille fichier lu: 1059314 ; Nb read: 11 ; Nb write:
11
~> tcopier1 0 fich3
blablabla
bliblibli
<touche ^D>
Taille tampon: 100000 ; Taille fichier lu: 20 ; Nb read: 2 ; Nb write: 2
~> cat fich3
# On doit trouver :
blablabla
bliblibli
```

2. Recopie de fichier version 2

Réécrire une version 2 du programme précédent, mais en utilisant les macros de Rousseau pour le traitement des erreurs. Pour cela, utiliser le *Makefile* fourni, construit préalablement par le script configure. Ce "*Makefile*" fournit une cible principale qui construit un programme exécutable écrit

en C en incorporant les fichiers de bibliothèque de Stevens. Il faut aussi que votre programme commence par la ligne magique

```
#include <stevens.h>
```

car ce fichier d'en-tête définit (entre autre) les macros de Rousseau, et réalise toutes les inclusions de fichiers d'en-tête nécessaires aux TPs.
Cette deuxième version utilisera un tampon de 10 000 octets seulement. On refera les mêmes tests que pour la version précédente et on comparera les résultats avec ceux obtenus précédemment.

3. Fichiers à trous

Ecrire un programme **write_a_hole** qui crée un fichier de nom "*fichier-troué*" qui contient une ligne de texte, un trou de 100 000 octets puis une ligne de texte. (Rappel: le trou s'obtient en déplaçant le curseur d'écriture au delà de la fin de fichier).
Exécuter le programme **write_a_hole**. Constatez que le fichier "*fichier-troué*" a bien une taille de plus de 100 000 octets. Effacer le fichier "fich1" des questions précédentes (dont la taille dépasse 1Mo). Avec les commandes :

```
~> du .
~> ls -ls fichier-troué
```

Exécuter le programme write_a_hole . Constatez que le fichier "fichier-troué" occupe moins de 100 blocs de 1024 octets[2].

Remarque importante :
Pour avoir un fonctionnement du système de fichiers non biaisé par le montage NFS (montage d'un disque sur une machine à travers le réseau), il faut travailler sur un disque local à la machine utilisée. Dans le cas des machines du MIPS, cela peut s'obtenir en créant un répertoire du nom de l'utilisateur sous "/tmp :"

```
~> mkdir /tmp/user
~> cd /tmp/user
```

Les tests d'exécution se feront sous ce répertoire. **Ne pas oublier en fin de séance de récupérer dans son espace utilisateur les sources des programmes développés et d'effacer ensuite "/tmp/user".**

4. Pour les plus courageux...

Ecrire une version **tcopier3** de **tcopier2** qui détecte les trous dans le fichier source et les recrée dans le fichier destination avec la primitive *lseek(2)*. Ceci permet d'éviter d'allouer des blocs pleins de 0 dans le fichier destination et gagne ainsi de la place sur le disque.

[2] Ce n'est pas le cas sous Cygwin qui est implémenté au dessus de Windows, car le système de fichier de Windows ne permet pas de traiter cela correctement.

Un bloc plein de zéro est une suite de BLKSIZE octets nuls dans le fichier source, dont l'adresse du premier octet est un multiple de BLKSIZE. BLKSIZE est la taille d'un bloc d'allocation du système de fichier et varie selon le type de système. On peut connaître la taille en utilisant la primitive *fstat* (cf. voir le manuel de *fstat(2)*) ou la commande *stat(1)*.

Utilisation de l'environnement des Tps sur son ordinateur personnel

Les consignes qui suivent sont à réaliser en dehors des séances de TP. Pour installer sur votre ordinateur l'environnement utilisé pour les TPs de cet enseignement, procédez comme suit. Créez un répertoire "*l3i6*" sur votre machine sous votre racine personnelle, comme pour les machines du MIPS. Placez sous "*l3i6*" une copie du répertoire "*~lahire/l3i6/lib*" du MIPS. (Pour la récupération, utilisez une archive intermédiaire en format .tgz qui préserve les permissions et les dates). Supprimez sous ce répertoire "*lib*" les sous-répertoires qui ne correspondent pas à votre machine (pour gagner de la place).
Ensuite, créez sous "*l3i6*" des répertoires de séance comme sur les machines du MIPS ("*01.entrees-sorties*",etc.), et dépliez dans ces répertoires les fournitures données pour les TPs. Vous devez juste mettre le script **configure** en accord avec l'emplacement du répertoire "*lib*", en adaptant le script **configure**. Si vous récupérez les archives de fourniture proprement, vous constaterez que les scripts configure sont des liens symboliques vers un seul et même fichier. Il suffit de modifier cet unique fichier pour que tous vos scripts configure fonctionnent. Vous pouvez aussi placer ce script adapté et unique dans votre répertoire *bin* personnel.
En fait l'environnement fourni se réduit à un fichier d'en-tête "*stevens.h*" et à une bibliothèque "*libstevens.a*" adaptée à votre machine. Vous pouvez adapter cette bibliothèque en la recompilant, car les sources sont fournis. Vous verrez dans le cours « Environnement de programmation », comment construire une bibliothèque.