TP Tube Processus

Systèmes d'Exploitation

2019-2020

Introduction

Il existe une solution simple qui permet de lancer des commandes dans un Shell depuis un programme `C` et de récupérer le résultat dans le même progamme C.

Cette fonctionnalité est possible grâce à la fonction `popen()` qui possède la signature suivante:

```
FILE *popen ( char *command, char *type);
```

L'argument `command` contient la commande à exécuter dans un Shell et `type` précise si nous souhaitons transmettre des données ou si nous souhaitons recevoir des données.

La fonction retourne en cas de succès un flux de données. Ce flux doit être fermé par la fonction `pclose()`

Que fait popen() ?

La fonction popen fait partie de la bibliothèque standard et va créer un tube de communication *half-duplex*. En interne, `popen` utilise un `pipe()` et réalise un `fork/exec` pour exécuter le programme shell qui va à son tour lancer la commande de l'utilisateur.

La direction du tube est déterminée par le deuxième argument, "type", qui peut prendre deux valeurs: "r" ou "w", pour "lire" ou "écrire".

Les tubes créés avec `popen()` doivent être fermés avec `pclose ()`.

Mise en application

Dans un premier temps, consultez le manunel de la commande Shell `sort`

```
man sort
```

Vous pouvez ensuite essayer cette commande avec :

```
printf " un\n deux\n trois\n" | sort
```

Nous allons à présent intéragir avec `sort` en utilisant un tube processus.

```
#include <stdio.h>
#define MAXSTRS 5
int main(void)
{
        int  cntr;
        FILE *pipe_fp;
        char *strings[MAXSTRS] = { "gamma", "bravo", "alpha",
                                   "charlie", "delta"};

        /* creation d un tube process pour utiliser la commande sort*/
        if (( pipe_fp = popen("sort", "w")) == NULL)
        {
                perror("popen");
                exit(1);
        }

        for(cntr=0; cntr<MAXSTRS; cntr++) {
                fputs(strings[cntr], pipe_fp);
                fputc('\n', pipe_fp);
        }

        pclose(pipe_fp);

        return(0);
}
```

Exercice

Nous souhaitons écrire un programme C qui va lire dans un fichier texte des expressions arithmétiques et les évaluer en utilisant le programme `bc`

Comprendre le programme `bc`

Consultez le manuel de la commande `bc`

```
man bc
```

Essayez la commande suivante :

```
echo "(3*4)/6" | bc
```

Etape 1

Pour cette étape nous allons simplement ouvrir un fichier qui contient une suite de lignes pour les afficher.

```c
#include <stdio.h>
#define USAGE "usage: %s <fichier des expressions>"

#define MAXLIGNE 80

int main(int argc, char** argv)
{
    char ligne[MAXLIGNE];
    if(argc < 2)
    {
        printf(USAGE,argv[0]);
        return(1);
    }
    FILE* input = fopen(argv[1],"r");
    if(input == NULL)
    {
        printf("Erreur ouverture du fichier %s",argv[1]);
        exit(1);
    }

    while ( fgets ( ligne, MAXLIGNE, input ) != NULL )
    {
        printf(ligne);
    }
    return(0);
}
```

Etape 2

Dans l'étape 2 notre objectif est de créer les chaines de caractères qui vont représenter les commandes Shell. Par exemple, si la ligne lue sur le fichier est `3+4` alors la commande va être: `echo '3+4' | bc`

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define USAGE "usage: %s <fichier des expressions>"
#define MAXLIGNE 80
#define MAXCMD 120
```

```c
int main(int argc, char** argv)
{
    char ligne[MAXLIGNE];
    char cmd[MAXLIGNE];

    if(argc < 2)
    {
        printf(USAGE,argv[0]);
        return(1);
    }

    FILE* input = fopen(argv[1],"r");
    if(input == NULL)
    {
        printf("Erreur ouverture du fichier %s",argv[1]);
        exit(1);
    }

    while ( fgets ( ligne, MAXLIGNE, input ) != NULL )
    {
        //ici nous retirons le retour a la ligne
        if( ligne[strlen(ligne)-1] == '\n')
        {
            ligne[strlen(ligne)-1] = '\0';
        }
        sprintf(cmd,"echo '%s' | bc",ligne);
        printf("%s\n",cmd);
    }
    return(0);
}
```

Etape 3

Maintenant que nous disposons des commandes Shell, nous allons utiliser un tube process et recupérer le résultat.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define USAGE "usage: %s fichier"
#define MAXLIGNE 80
#define MAXCMD 120
#define MAXRESULT 80
```

```c
int main(int argc, char** argv)
{
    char ligne[MAXLIGNE];
    char cmd[MAXLIGNE];
    char resultat[MAXRESULT];

    if(argc < 2)
    {
        printf(USAGE,argv[0]);
        return(1);
    }

    FILE* input = fopen(argv[1],"r");
    if(input == NULL)
    {
        printf("Erreur ouverture du fichier %s",argv[1]);
        exit(1);
    }

    while ( fgets ( ligne, MAXLIGNE, input ) != NULL )
    {
        //ici nous retirons le retour a la ligne
        if( ligne[strlen(ligne)-1] == '\n')
        {
            ligne[strlen(ligne)-1] = '\0';
        }
        sprintf(cmd,"echo '%s' | bc",ligne);

        FILE* pipe_fp;
        /* creation d un tube process pour utiliser la commande sort*/
        if (( pipe_fp = popen(cmd, "r")) == NULL)
        {
                perror("popen");
                exit(1);
        }
        if ( fgets(resultat, MAXRESULT, pipe_fp) == NULL){
                perror("fgets");
                exit(1);
        }
        pclose(pipe_fp);
        printf("%s",resultat);
    }
    return(0);
}
```