STI 1ère année – Programmation Système

TD 8: shell et redirections

J. Briffaut

1 Processus

1.1 Création de processus : Mutation et Redirection

Exercice 1 Créer un programme C qui va simuler un mini-shell avec la gestion des redirection vers un fichier : >.

Une fois lancé, ce programme attendra sur l'entrée standard que l'on entre une commande. Cette commande sera ensuite exécutée, le processus père de votre mini-shell attendra alors que le processus finisse avant de demander à nouveau d'entrée un commande. Si la commande correspond a `exit`, votre shell s'arrête. Si il y a un problème lors de l'exécution de la commande, vous afficherez l'erreur correspondante.

Si l'utilisateur entre une commande avec une redirection vers un fichier (>), la sortie de la commande sera renvoyée dans ce fichier.

Par exemple, l'exécution de la commande `/bin/echo toto` affichera toto à l'écran, alors que `/bin/echo toto > test` créera un fichier test qui contiendra la chaine toto.

Vous devrez utiliser les fonctions suivantes : `execv`, `dup2`.

Listing 1 – Solution de l'exercice 1

```
#include<stdlib.h>
#include<stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>
#include<unistd.h>


#define BUFFER 512
#define DEBUG 1

int main() {
        char command[BUFFER];
        char fdest[BUFFER];
        char* arg[BUFFER];
        int nb_args,i;
        while(1){
                fdest[0]='\0';
                //on rÃľcupÃíre une commande a exÃľcuter
                nb_args=read_command(command, &arg, fdest);
                arg[0]=command;
                //si la commande correspond a exit, on quitte
                if(nb_args==-1){
                        printf("Exit");
                        exit(EXIT_SUCCESS);
                }

                if(DEBUG){
                        printf("commande a exÃľcuter: %s, %d arguments [",command,nb_args);
                        for(i=0;i<nb_args;i++)
                                printf("%s , ",arg[i]);
                        printf("], fdest : %s\n",fdest);
                }

                //on execute la commande
```

```
                run_command(command,arg,fdest);
        }
}

int read_command(char* command, char* arg[], char * fdest){
        int cpta=1;
        int c,i;
        char * chaine;
        // lire la command
        scanf("%s",command);
        if (strcmp(command,"exit")==0) {
                return -1;
        }

        //lire les arguments
        while((c=getchar())!='\n'){
                chaine=malloc((BUFFER-1) * sizeof(char));
                scanf("%s",chaine);
                if (strcmp(chaine,">")==0) {
                        //lire le fichier de sortie
                        scanf("%s",fdest);
                        break;
                }
                arg[cpta++]=chaine;
        }

        //on retourne le nombre d'arguments
        return cpta;
}

int run_command(char * command, char * arg[], char * fdest) {
        pid_t status;
        int fd;
        status = fork();
        switch (status) {
                case -1 :
                        perror("Creation processus");
                        exit(EXIT_FAILURE);
                case 0 :
                        // fichier de sortie ?
                        if(strlen(fdest)>0){
                                fd=open(fdest, O_WRONLY | O_CREAT | O_TRUNC, 0666);
                                dup2(fd, STDOUT_FILENO);
                        }
                        //utilisation de execv pour exÃľcuter la commande
                        if(execv(command,arg)==-1){
                                perror("Erreur : ");
                                exit(EXIT_FAILURE);
                        }
                        if(strlen(fdest)>0){
                                close(fd);
                        }
                        exit(EXIT_SUCCESS);
                        break;
                default:
                        //attente de la fin de la commande
                        wait(&status);
                        if (WIFEXITED(status))
                                printf("commande executÃľe sans erreur\n");
                        else {
                                printf("commande executÃľe avec erreur\n");
                        }

        }
}
```