TP6 Système d'exploitation

Antoine Maendly

Sources

- https://www.youtube.com/watch?v=83M5-NPDeWs
- https://www.youtube.com/watch?v=_5SCtRNnf9U&t=407s
- Slides Exo

Architecture

1. Module builtin

```
int cd(char *dst);

void exit_command(int *background_pid, int *foreground_pid);

void prompt(void);
```

Avec trois function qui permette de changer de working directory, de quitter le shell et d'afficher le prompt avec le working directory actuel

2. Module argument

```
char** parsing(char *line);

int count_arg(char **parsed);
```

Avec la fonction parsing qui renvoie un tableau de string à partir de l'entrée de l'utilisateur et la fonction count_arg qui permet simplement de donné le nombre d'arguments qu'il y a dans ce tableau

3. Module jobs

```
void handle_background_job(char **parsed, int arg_count, int *background_pid);

void handle_foreground_job(char **parsed, int *foreground_pid, struct sigaction *act);
```

Et finalement deux fonction qui permette de gérer et lancer les jobs de fond et de premier plan

Makefile

```
CC = gcc
MODULE1 = builtin
MODULE2 = argument
MODULE3 = jobs

all: shell $(MODULE1) $(MODULE2)

clean:
    rm shell $(MODULE1) $(MODULE2) $(MODULE3)

shell: shell.c $(MODULE1).o $(MODULE2).o $(MODULE3).o
    $(CC) -o shell shell.c $(MODULE1).o $(MODULE2).o $(MODULE3).o

$(MODULE1): $(MODULE1).c $(MODULE1).h
    $(CC) -c $(MODULE1).c -o $(MODULE1).o

$(MODULE2): $(MODULE2).c $(MODULE2).h
    $(CC) -c $(MODULE2).c -o $(MODULE2).o

$(MODULE3): $(MODULE3).c $(MODULE3).h
    $(CC) -c $(MODULE3).c -o $(MODULE3).o
```

Background job et foreground

Pour gérer le background job, j'ai définie une variable globale pour le foreground job et le background job qui vaut 0 si il n'y a pas de job en cours et le pid de l'enfant si un job est en cours. Cela me permet de checker depuis le sig_handler si le signal provient bien du background job. J'utilise le pid du foreground job

pour m'assurer de bien terminer ses proccessus lors de la fermeture ou bien si le shell recois un signal SIGHUP. De plus, il me permet de rediriger un SIGINT, si jamais il y a un job en premier plan.

Programme background_program

J'ai aussi créé un petit programme affiche juste qu'il tourne en fond pour pouvoir voir l'effet des différents signaux

Quelque tests

Les quelques tests que j'ai pu effectué sont :

- Envoyer un SIGHUP avec un programme en fond et un en premier plan depuis un autre shell
- Envoyer un SIGINT avec un programme en fond et en premier plan depuis un autre shell
- Lancé des programmes sleep avec des temps différents pour voir les différents comportements lié au sig_handler et au wait/waitpid

(Conclusion)

Je ne crois pas qu'il y ait trop de bug ou de comportement inattendu mais étant donné que je suis sur mac il pourrait y avoir quelque problème à ce niveau la. Par exemple, mon test pour la commande cd ne marche pas car, sur macos, le /var/... est directement redirigé vers /private/var/....

Vrai conclusion

Finalement, j'ai du faire plus de modification que prévu lorsque je le fais tourné sur linux (Ubuntu).

- Définir #define _GNU_SOURCE sinon des structures comme siginfo_t ou des macros ne sont pas définis
- Modification de la commande parsed car j'avais des problèmes de segmentation fault. Au lieu de realloc à chaque fois qu'on parse un argument, je définie une capacité max de 19 arguments. Je realloc que si on dépasse la limite, et je double la capacité. Ceci a résoud le problème
- La commande cd n'a pas changer mais fonctionne comme prévu sur linux