Kadri Yanis

Programmation système : Réalisation d'un Shell miniature

Cahier des charges

Utilisation du langage C

Créer un Shell miniature Bash (minishell)

Plusieurs contributions/concepts de programmation système

- Exécuter des commandes internes (cd, pwd, mkdir,...)
- Exécuter des commandes externes avec par le marqueur '!' (ls, cp, rm, ...)
- Exécuter des commandes externes en arrière plan (de manière asynchrone) et réaliser la commande jobs
- Effectuer les redirections

Utilisation de signaux et de processus

Utilisation obligatoire de la fonction exec et non de la fonction system

Résultat de la mission

```
* Shell started
1: /home/ykadri/Bureau/CoursBUTS3/ProgSysteme/SAE/projetMinisGitMarche/projetminishell/minishell> cd ..
2: /home/ykadri/Bureau/CoursBUTS3/ProgSysteme/SAE/projetMinisGitMarche/projetminishell> cd minishell
3: /home/ykadri/Bureau/CoursBUTS3/ProgSysteme/SAE/projetMinisGitMarche/projetminishell/minishell> ! ls
main.c  main.o  Makefile  myshell  Shell.c  Shell.h  Shell.o  StringVector.c  StringVector.h  StringVector.o
4: /home/ykadri/Bureau/CoursBUTS3/ProgSysteme/SAE/projetMinisGitMarche/projetminishell/minishell> mkdir newFile
Directory newFile has been created
5: /home/ykadri/Bureau/CoursBUTS3/ProgSysteme/SAE/projetMinisGitMarche/projetminishell/minishell> ! ls
main.c  main.o  Makefile  myshell  newFile  Shell.c  Shell.h  Shell.o  StringVector.c  StringVector.h  StringVec
6: /home/ykadri/Bureau/CoursBUTS3/ProgSysteme/SAE/projetMinisGitMarche/projetminishell/minishell> ! xeyes &
[8575] xeyes
7: /home/ykadri/Bureau/CoursBUTS3/ProgSysteme/SAE/projetMinisGitMarche/projetminishell/minishell> jobs
Processus : 8575
8: /home/ykadri/Bureau/CoursBUTS3/ProgSysteme/SAE/projetMinisGitMarche/projetminishell/minishell> jobs
Processus 8575 : fin
9: /home/ykadri/Bureau/CoursBUTS3/ProgSysteme/SAE/projetMinisGitMarche/projetminishell/minishell> ! xeyes &
[8723] xeyes
10: /home/ykadri/Bureau/CoursBUTS3/ProgSysteme/SAE/projetMinisGitMarche/projetminishell/minishell> ! xeyes &
[8772] xeyes
11: /home/ykadri/Bureau/CoursBUTS3/ProgSysteme/SAE/projetMinisGitMarche/projetminishell/minishell> jobs
Processus : 8723
Processus : 8772
12: /home/ykadri/Bureau/CoursBUTS3/ProgSysteme/SAE/projetMinisGitMarche/projetminishell/minishell> jobs
Processus 8772 : fin
Processus 8723 : fin
13: /home/ykadri/Bureau/CoursBUTS3/ProgSysteme/SAE/projetMinisGitMarche/projetminishell/minishell> ! xeyes
14: /home/ykadri/Bureau/CoursBUTS3/ProgSysteme/SAE/projetMinisGitMarche/projetminishell/minishell> jobs
15: /home/ykadri/Bureau/CoursBUTS3/ProgSysteme/SAE/projetMinisGitMarche/projetminishell/minishell> pwd
/home/ykadri/Bureau/CoursBUTS3/ProgSysteme/SAE/projetMinisGitMarche/projetminishell/minishell
16: /home/ykadri/Bureau/CoursBUTS3/ProgSysteme/SAE/projetMinisGitMarche/projetminishell/minishell> ! xeyes
```

```c
/// @brief Méthode qui permet d'éxécuter des commandes externes (en arrière plan ou en avant plan)
/// @param this Structure du shell
/// @param args Structure qui contient la ligne de commande mise et d'autre information
static void
do_system( struct Shell *this, const struct StringVector *args )
{
    //variables permettant de gérer la présence de & (cas par défaut = sans &)
    int taille;
    bool aEsperluette;
    //on modifie les variables si la commande contient un &
    if (*string_vector_get(args, args->size-1) == '&'){
        aEsperluette = true;
        taille = args->size-1;
        //On fait un signal quand le fils à fini son execution
        struct sigaction sa;
        //On définie la fonction à appeller lorsque l'éxecution d'un fils sera fini
        sa.sa_handler = fin_fils;
        //bloque des signaux spécifiques
        sigemptyset( &sa.sa_mask );
        //Défini un flag pour évité qu'il y ai d'autres signaux qui se lancent en même temps
        sa.sa_flags = SA_RESTART;
        //On définie le signal lorque le fils aura fini de s'éxecuter
        int valeurRetour = sigaction( SIGCHLD, &sa, NULL );
        if ( valeurRetour < 0 ) {
            perror( "Echec du signal" );
            exit( EXIT_FAILURE );
        }
    }
    else {
```

Méthode

Utilisation papier pour les algorithmes

Échanges de points vus

Travail en équipe : partage des tâches

Compétence acquises

Savoir comment fonctionne un shell

Savoir analyser une ligne de commande mise en entrée (concernant les algorithmes pour les commandes interne, externes, etc) et l'exécuter

Amélioration de la propreté et lisibilité du code

Amélioration de la connaissance et de la pratique sur le langage C et de ses fonctionnalités à l'aide des bibliothèques système (signal, wait, open, exec, write, read, etc.) ou d'autres (pointeurs, gestion de la mémoire, ...)

Technologie (langages et application)

C

Visual Studio Code