IN405 - Système d'exploitation

S. Gougeaud

2017/2018

Feuille de Travaux dirigés 1

Terminal – Shell

Exercice 1 – Compréhension des commandes de base

Soit la liste de commandes suivante : `cat cd cp diff echo gcc gdb ls make man mkdir mv rm rmdir sudo tar time touch vi`

1. Donnez une brève description pour chacune des commandes.
2. Quelles commandes consistent en l'exécution d'un binaire ?
3. Quels chemins sont représentés par les symboles suivants : `.`, `..`, `~`.

Exercice 2 – Première utilisation du terminal

Pour chacune des questions suivantes, exécutez la commande correspondante.

1. Déplacez vous dans le répertoire temporaire de votre système de fichiers.
2. Créez le répertoire `project` ainsi que les sous-répertoires `doc`, `include` et `src`.
3. Au sein du dossier `project`, créez un fichier `README` contenant votre nom et prénom. Créez le fichier `func.h` dans `include`, les fichiers `main.c` et `func.c` dans `src`.
4. Affichez la hiérarchie complète du répertoire `project` et des ses sous-répertoires, puis écrivez ce résultat dans `contents.txt`.
5. Créez une copie du répertoire `project` que vous nommerez `projectV2`. Supprimez le répertoire `project`.
6. Créez l'archive `pv2.tar` contenant l'ensemble du répertoire `projectV2`.

Exercice 3 – Premier script Shell

Afin d'automatiser l'exécution de commandes (comme par exemple la compilation d'un projet ou l'exécution d'un jeu de tests), il est possible de les rassembler dans un fichier. Ce type de fichier est appelé script. Placez l'ensemble des commandes écrites dans l'exercice 2 dans un script Shell, et exécutez-le. Le résultat est-il le même que dans l'exercice 2 ?

Exercice 4 – Shell en C

A l'aide de la fonction `system`, faites un programme C affichant le contenu de votre répertoire personnel.

Exercice 5 – Débogage

Compilez le programme "Code mystère" en utilisant l'option `-g` de `gcc`, puis déboguez-le à l'aide de `gdb` jusqu'à atteindre l'exécution normale du programme.

Rappel des commandes gdb :
`breakpoint fichier:ligne` – *ajout d'un point d'arrêt dans le code*
`run arg1 arg2 ...` – *exécution du programme*
`CTRL + c` – *envoi d'un signal d'interruption au programme*
`next` – *exécution de l'instruction suivante*
`continue` – *reprise de l'exécution du programme*
`print var` – *affichage du contenu d'une variable*
`backtrace` – *affichage de la pile d'appels des fonctions*
`up/down i` – *remontée/descente de i dans la pile d'appels*
`quit` – *arrêt du débogueur*

Exercice 6 – Optionnel : Shell en C (bis)

Écrivez le programme C répondant aux questions de l'exercice 2.

```c
#include <stdio.h>
#include <stdlib.h>
#define NUM_STEPS 8

typedef struct {double r; double i;} complex_t;

int main (int argc, char ** argv) {
   int i;
   complex_t * complexArray;
   complex_t oddSum = {.0, .0}, evenSum = {.0, .0};

   complexArray = malloc (NUM_STEPS * sizeof (complex_t));

   for (i = 0; i != NUM_STEPS; ++i) {
      complexArray [i] .r = i + 4;
      complexArray [i] .i = 3 * i;
   }

   for (i = 0; i != NUM_STEPS; i += 2) {
      evenSum.r += complexArray [i] .r;
      evenSum.i += complexArray [i] .i;
   }

   for (i = 1; i != NUM_STEPS; i += 2) {
      oddSum.r += complexArray [i] .r;
      oddSum.i += complexArray [i] .i;
   }

   printf ("Sums are: (%f, %f) & (%f, %f)\n",
      evenSum.r, evenSum.i, oddSum.r, oddSum.i);

   return 0;
}
```

Figure 1.1: Code mystère