Système M1 — TP 3 : Shell

Semaine du 5 octobre 2009

Exercice 1 – *Shell 1*

Écrire une fonction qui effectue la boucle suivante : un prompt attend une commande avec éventuellement des arguments, exécute la commande, et enfin affiche la valeur de sortie du processus. Exemple :

```
[rao@liafa0t test]$ ../shell1
>ls
tata  toto
Valeur de retour: 0
>ls -la
total 12
drwxr-xr-x 2 rao ater 4096 Oct  6 16:45 .
drwxr-xr-x 3 rao ater 4096 Oct  6 16:47 ..
-rw-r--r-- 1 rao ater    5 Oct  6 16:45 tata
-rw-r--r-- 1 rao ater    0 Oct  6 16:45 toto
Valeur de retour: 0
>
```

Indication Vous pouvez utiliser la fonction `char **parse(char *s,char *c)` qui prend deux chaînes de caractères `s` et `c`, et renvoie un tableau de toutes les chaînes non vides de `s` séparées par un(ou plusieurs) caractère(s) de `c`. Le tableau sera terminé par le pointeur `NULL`.

```
#include <string.h>
char **parse(char *s,char *c)
{
  char ** r, * tmp, * save;
  int i,j;
  for(i=0,j=0; s[j]; j++)
    if((!index(c,s[j])) && (j==0 || index(c,s[j-1]))) i++;
  r=(char**)malloc((i+1)*sizeof(char*));
  for (i=0,tmp=s; ;i++,tmp=NULL){
    r[i]=strtok_r(tmp,c,&save);
    if (r[i] == NULL) break;
  }
  return r;
}
```

Par exemple, `parse("le porte-manteau"," -")` donnera `{"le","porte","manteau",NULL}`.

Exercice 2 – *Tube 1*

Écrire une fonction `void execute_tube(char **a,char **b)` prenant deux tableaux de chaînes de caractères terminés par `NULL`, chaque tableau correspondant à une commande avec ses arguments. La fonction exécute les deux commandes, en redirigeant la sortie de la première commande sur l'entrée de la seconde commande. Exemple :

```
void main() {
  char *cmd1[]={"ls","-la",NULL};
  char *cmd2[]={"grep","toto",NULL};
  execute_tube(cmd1,cmd2);
}
```

affichera :

```
[rao@liafa0t test]$ ../cmdtube2
-rw-r--r-- 1 rao ater    0 Oct  6 16:45 toto
```

Exercice 3 – *Tube 2*

Écrire une fonction `void execute_tube_va(char **,...)` à nombre variable d'arguments, prenant une liste de tableaux de chaînes de caractères terminés par `NULL`, chaque tableau correspondant à une commande avec ses arguments. La fonction redirigera la sortie de la première commande sur l'entrée de la seconde commande, et ainsi de suite.

```
void main() {
  char *cmd1[]={"ls","-la",NULL};
  char *cmd2[]={"grep","toto",NULL};
  char *cmd3[]={"sort",NULL};
  execute_tube_va(cmd1,cmd2,cmd3,NULL);
}
```

Exercice 4 – *Shell 2*

Écrire un programme qui effectue la boucle suivante : un prompt attend des commandes (éventuellement avec des arguments) séparées par des `|`, et exécute les commandes en redirigeant la sortie de la première commande sur l'entrée de la seconde commande, et ainsi de suite. Exemple :

```
[rao@liafa0t test]$ ../shell2
>ls -l
total 4
-rw-r--r-- 1 rao ater 5 Oct  6 16:45 tata
-rw-r--r-- 1 rao ater 0 Oct  6 16:45 toto
Valeur de retour: 0
>ls -la | sort -r
total 12
-rw-r--r-- 1 rao ater    5 Oct  6 16:45 tata
-rw-r--r-- 1 rao ater    0 Oct  6 16:45 toto
drwxr-xr-x 3 rao ater 4096 Oct  6 17:29 ..
drwxr-xr-x 2 rao ater 4096 Oct  6 16:45 .
Valeur de retour: 0
Valeur de retour: 0
```