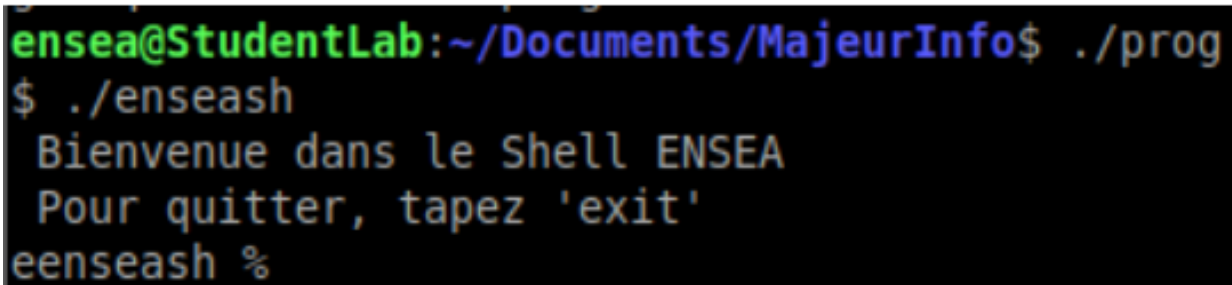


# Micro Shell

1) Display of a welcome message, followed by a simple prompt

```
1  #include "question1.h"
2
3  int main(void)
4  {
5      int fd = 1;
6      int size = 32;
7      write(fd, "$ ./enseash", 12);
8      write(fd, "\n Bienvenue dans le Shell ENSEA", size);
9      write(fd, "\n Pour quitter, tapez 'exit' \n", size);
10     write(fd, "enseash % \n", 12);
11     return 0;
12 }
13
```

We get the following result :



```
ensea@StudentLab:~/Documents/MajeurInfo$ ./prog
$ ./enseash
 Bienvenue dans le Shell ENSEA
 Pour quitter, tapez 'exit'
enseash %
```

We display our welcome message well.

## 2) Executing the entered command and returning to the prompt (REPL)

```
3  int main(void)
4  {
5      int fd = 1;
6      int nb_octets = 32;
7      char cmd[50] = {0};
8      ssize_t size;
9
10     write(fd, "$ ./enseash", 12);
11     write(fd, "\n Bienvenue dans le Shell ENSEA.", nb_octets);
12     write(fd, "\n Pour quitter, tapez 'exit'.\n", nb_octets);
13
14     while(1){
15         size = read(fd, cmd, 50); // Ecoute d'une commande
16         char fortune[size];
17         strncpy(fortune, cmd, size-1); // Copie les size premiers caractères de la commande
18                                     // dans le tableau fortune
19         if(strcmp(fortune, "fortune") == 0){ // Gestion de la commande fortune
20             write(fd, "Today is what happened to yesterday.\n", 39);
21             return 1;
22         }
23         else if(strcmp(cmd, "exit") == 0){ // Compare la chaine "exit" avec la commande écrite
24             printf("\t EXIT \n");
25             return 1;
26         }
27         else{ // Gestion de l'erreur
28             write(fd, "Non reconnu \n", 12);
29             write(fd, cmd, size);
30             return 1;
31         }
32     }
33     return 0;
34 }
35
```

We get the following result when we enter the command "fortune" :

```
ensea@StudentLab:~/Documents/MajeurInfo$ ./prog
$ ./enseash
Bienvenue dans le Shell ENSEA.
Pour quitter, tapez 'exit'.
ffortune
Today is what happened to yesterday.
```

We read the command "fortune" which executes the desired script.

### 3) Handling shell exit with the "exit" command or a <ctrl>+d

```
1  #include "question1.h"
2
3  int main()
4  {
5      int fd = 1;
6      int nb_octets = 32;
7      char cmd[50] = {0};
8      pid_t pid;
9      ssize_t size;
10     int status;
11
12     write(fd, "$ ./enseash\n", 12);
13     write(fd, "\n Bienvenue dans le Shell ENSEA.", nb_octets);
14     write(fd, "\n Pour quitter, tapez 'exit'.\n", nb_octets);
15     write(fd, "enseash % ", 12);
16
17     while(1){
18         size = read(fd, cmd, 50); // Ecoute d'une commande
19         cmd[size-1]='\0'; // Initialise la commande
20
21         if(strcmp(cmd, "exit") == 0){ // Compare la chaine "exit" avec la commande écrite
22             write(fd, "\n Bye bye ... \n", 15);
23             break; // Sort au cas où la commande exit a été entrée
24         }
25
26         else{
27             pid = fork(); // Fork
28             if(pid == 0){ // Code du fils
29                 cmd[size-1] = '\0'; // Réinitialise la commande
30                 execlp(cmd, cmd, NULL); // Execute les commandes du shell grâce à
31                                         // l'executable situé dans la variable
32                                         // d'environnement PATH
33
34                 if(strcmp(cmd, "fortune") == 0){ // Gestion de la commande fortune
35                     write(fd, "Today is what happened to yesterday.\n", 39);
36                     return 1;
37                 }
38             }
39             else if (pid > 0){ // Code du père
40                 wait(&status); // Attend que le fils ait fini son processus avant de commencer
41                 write(fd, "\n", 2);
42             }
43         }
44     }
45     return 0;
46 }
```

We get the following result when we enter the "exit" command:

```
ensea@StudentLab:~/Documents/MajeurInfo$ ./prog
$ ./enseash

Bienvenue dans le Shell ENSEA.
Pour quitter, tapez 'exit'.
enseash % eexit
Bye bye ...
fensea@StudentLab:~/Documents/MajeurInfo$
```

We exit the shell with the "exit" command.

#### 4) Display the return code (or signal) of the previous command in the the prompt

```
3 int main()
4 {
5     int fd = 1;
6     char cmd[50] = {0};
7     pid_t pid;
8     ssize_t size;
9     int status;
10    int i = 0;
11    char msg[20];
12
13    write(fd, "$ ./enseash", 12);
14    write(fd, "\n Bienvenue dans le Shell ENSEA.", 33);
15    write(fd, "\n Pour quitter, tapez 'exit'.\n", 32);
16    write(fd, "enseash % ", 10);
17
18    while(1){
19        size = read(fd, cmd, 50); // Ecoute d'une commande
20        cmd[size-1] = '\0'; // Initialise la commande
21        i++;
22
23        if(strcmp(cmd, "exit") == 0){ // Compare la chaine "exit" avec la commande écrite
24            write(fd, "Bye bye ... \n", 14);
25            break; // Sort au cas où la commande exit a été entrée
26        }
27
28        else{
29            pid = fork(); // Fork
30            if (pid == -1){ // Gestion de l'erreur
31                perror("Fork impossible");
32                exit(EXIT_FAILURE);
33            }
34            else if(pid == 0){ // Code du fils
35                cmd[size-1] = '\0'; // Réinitialise la commande
36                execlp(cmd, cmd, NULL); // Execute les commandes du shell grâce à
37                                     // l'exécutable situé dans la variable
38                                     // d'environnement PATH
39
40                if(strcmp(cmd, "fortune") == 0){ // Gestion de la commande fortune
41                    write(fd, "Today is what happened to yesterday.\n", 39);
42                    return 1;
43                }
44            }
45            else{ // Code du père
46                wait(&status); // Attend que le fils ait fini son processus avant de commencer
47                sprintf(msg, "enseash [%s:%d] % ", cmd, i); // Ecris la commande précédente et le nombre de
48                write(fd, msg, strlen(msg)); // commandes déjà exécutées
49            }
50        }
51    }
52    return 0;
}
```

We get the following result when we run the program:

```
ensea@StudentLab:~/Documents/MajeurInfo$ ./prog
$ ./enseash
Bienvenue dans le Shell ENSEA.
Pour quitter, tapez 'exit'.
enseash % ls
enseash prog Q1.c Q2b.c Q2.h Q3.h Q4.h Q5.h question1.h TDm1
Makefile q1.c Q1.h Q2.c Q3.c Q4.c Q5.c question1.c question1.o TDm2
enseash [ls:1] pwd
/home/ensea/Documents/MajeurInfo
enseash [pwd:2] fortune
Today is what happened to yesterday.
enseash [fortune:3] ls
enseash prog Q1.c Q2b.c Q2.h Q3.h Q4.h Q5.h question1.h TDm1
Makefile q1.c Q1.h Q2.c Q3.c Q4.c Q5.c question1.c question1.o TDm2
enseash [ls:4] pwd
/home/ensea/Documents/MajeurInfo
enseash [pwd:5]
```

For each command execution, we obtain the previous command and the number of commands already executed.

## 5) Measuring the execution time of the command using the `clock_gettime`

```
15 write(fd, "$ ./enseash\n", 12);
16 write(fd, "\n Bienvenue dans le Shell ENSEA.", 33);
17 write(fd, "\n Pour quitter, tapez 'exit'.\n", 32);
18 write(fd, "enseash % ", 10);
19
20 while(1){
21     size = read(fd, cmd, 50); // Ecoute d'une commande
22     cmd[size-1] = '\0';      // Initialise la commande
23     i++;
24
25     if(strcmp(cmd, "exit") == 0){ // Compare la chaine "exit" avec la commande écrite
26         write(fd, "Bye bye ... \n", 14);
27         break; // Sort au cas où la commande exit a été entrée
28     }
29
30     else{
31         pid = fork(); // Fork
32         if (pid == -1){ // Gestion de l'erreur
33             perror("Fork impossible");
34             exit(EXIT_FAILURE);
35         }
36         else if(pid == 0){ // Code du fils
37
38             cmd[size-1] = '\0'; // Réinitialise la commande
39             clock_gettime( CLOCK_REALTIME, &start); // Lance le décompte
40             execlp(cmd, cmd, NULL); // Exécute les commandes du shell grâce à
41             clock_gettime( CLOCK_REALTIME, &stop); // Stoppe le décompte
42
43             time = (stop.tv_sec - start.tv_sec) + (stop.tv_nsec - start.tv_nsec) / 1000000; // Calcul le temps entre le début et la fin
44                                                     // du décompte en millisecondes
45
46             if(strcmp(cmd, "fortune") == 0){ // Gestion de la commande fortune
47                 write(fd, "Today is what happened to yesterday.\n", 39);
48                 return 1;
49             }
50         }
51         else{ // Code du père
52             wait(&status); // Attend que le fils ait fini son processus avant de commencer
53             sprintf(msg, "enseash [%s:%d]\n", cmd, i, time); // Ecrit la commande précédente, le nombre de
54                                                         // commandes déjà exécutées et le temps
55                                                         // d'exécution de la commande
56             write(fd, msg, strlen(msg));
57         }
58     }
59 }
60 return 0;
```

We tried to use the `clock_gettime()` function but unfortunately we were unable to measure the execution time of the command.

Here is also our `main.h` file with the libraries we used.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <sys/wait.h> // for wait
5 #include <sys/types.h> // for pid_t
6 #include <unistd.h> // for fork
7 #include <time.h> // for time
8
9 int main(void);
```