INSTITUT SUPERIEUR INFORMATIQUE ISI
المعهد العالي للإعلامية

RÉPUBLIQUE TUNISIENNE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR
ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITÉ DE TUNIS EL MANAR
INSTITUT SUPÉRIEUR D'INFORMATIQUE

جامعة تونس المنار
UNIVERSITÉ DE TUNIS EL MANAR

Rapport de Projet Système d'exploitation

Réalisé Par :

Aouadi Bayram
Belhadj Ali Mohamed Aymen

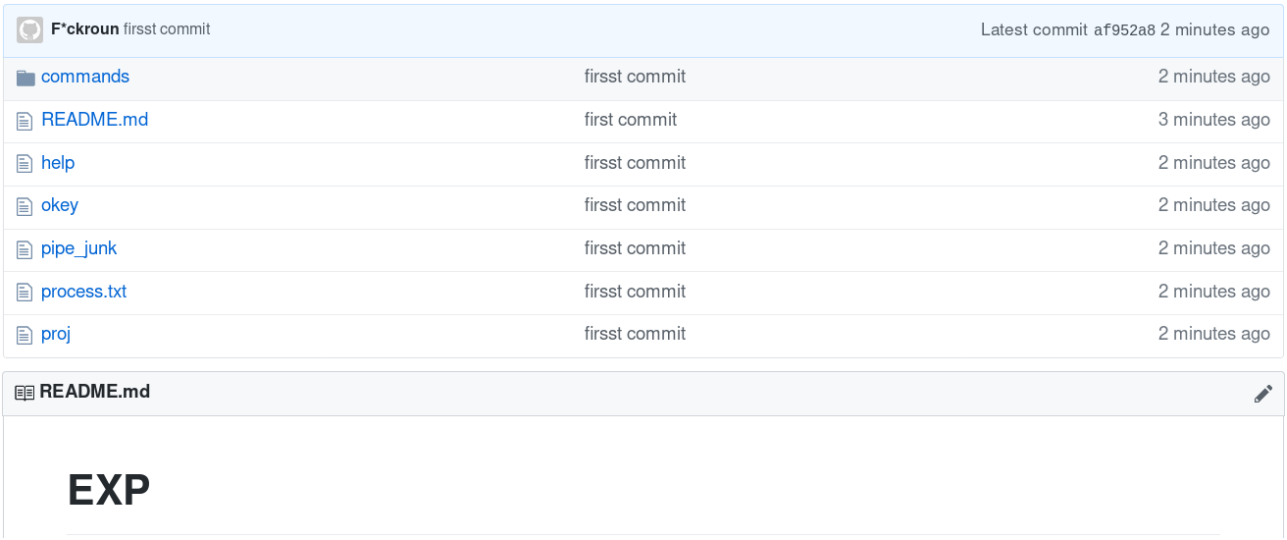
Classe : 1ere année ingénieur IDISC

Encadrante : Mme OUERGHI Hajer

Année Universitaire : 2019-2020

Partie I : Interpréteur de commande UNIX simplifié

Lien Github : https://github.com/WTFast69/EXP.git



Exemple d'execution :

pwd2 :

```
root@fuckroun:/root/Documents/exp/exp>>pwd2
/root/Documents/exp/exp
```

ls2 :

```
root@fuckroun:/root/Documents/exp/exp>>ls2
 okey  ll  .  .idea  ..  proj  process.txt  commands  pipe_junk  .git
```

touch2 :

```
root@fuckroun:/root/Documents/exp/exp>>ls2
 okey  .  .idea  ..  proj  process.txt  commands  pipe_junk  .git
root@fuckroun:/root/Documents/exp/exp>>touch2 lol

root@fuckroun:/root/Documents/exp/exp>>ls2
 okey  .  lol  .idea  ..  proj  process.txt  commands  pipe_junk  .git
```

mkdir2 :

```
root@fuckroun:/root/Documents/exp/exp>>ls2
 okey   .   .idea   ..   proj   process.txt   commands   pipe_junk   .git
root@fuckroun:/root/Documents/exp/exp>>mkdir2 lol
root@fuckroun:/root/Documents/exp/exp>>ls2
 okey   .   lol   .idea   ..   proj   process.txt   commands   pipe_junk   .git
```

cat2 :

```
root@fuckroun:/root/Documents/exp/exp>>cat2 process.txt
6, 2, 5
4, 3, 3
5, 3, 19
1, 6, 1
3, 7, 2
2, 8, 1
```

exit :

```
root@fuckroun:~/Documents/exp/exp# ./proj
Choose an option :
1 - Shell Command Line :
2 - Process Scheduler :
->>1
root@fuckroun:/root/Documents/exp/exp>>exit
Choose an option :
1 - Shell Command Line :
2 - Process Scheduler :
->>
```

HELP :

```
root@fuckroun:/root/Documents/exp/exp>>HELP


********** HELP MANUAL ************

cd2 : change current directory
alias : make an alias for a command
pwd2 : print the current directory's path
ls2 : list files and directories
cat2 : Display a file's content
rm2 : remove files or directories
mkdir2 : make a directory
touch2 : create a file
echo2 : print variable content
exit : exit command line
HELP : Show this manual

***********************************
```

alias :

```
root@fuckroun:/root/Documents/exp/exp>>alias kat=cat2
root@fuckroun:/root/Documents/exp/exp>>kat process.txt
6, 2, 5
4, 3, 3
5, 3, 19
1, 6, 1
3, 7, 2
2, 8, 1
```

echo2 :

```
root@fuckroun:/root/Documents/exp/exp>>echo2 qsdqsd
qsdqsd
```

cd2 :

```
root@fuckroun:/root/Documents/exp/exp>>cd2 SS
root@fuckroun:/root/Documents/exp/exp/SS>>cd2 ../..
```

Redirection entrée / sortie :

(>)

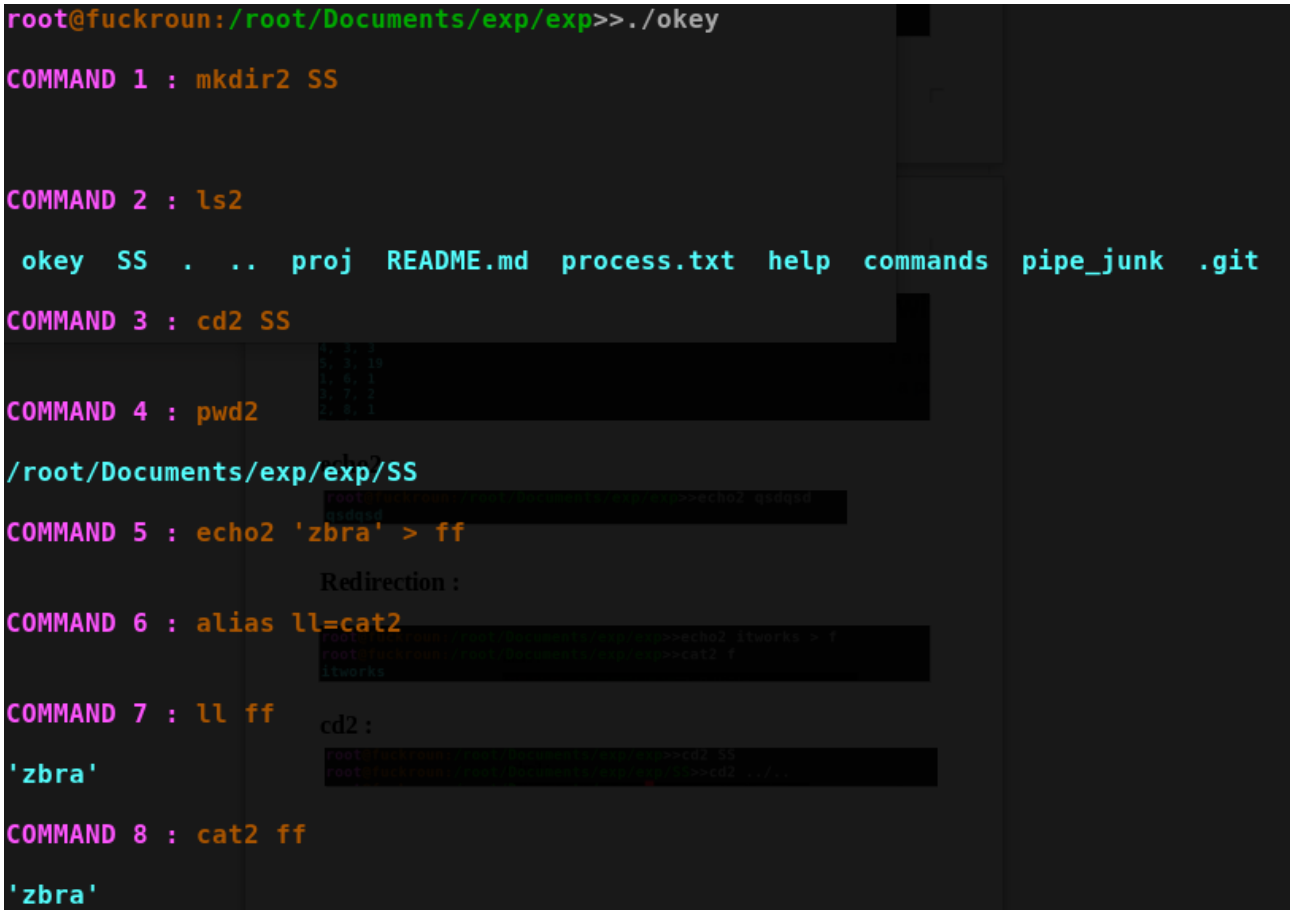
```
root@fuckroun:/root/Documents/exp/exp>>echo2 itworks > f
root@fuckroun:/root/Documents/exp/exp>>cat2 f
itworks
```

(<)

```
root@fuckroun:/root/Documents/exp/exp>>cat2 < process.txt
6, 2, 5
4, 3, 3
5, 3, 19
1, 6, 1
3, 7, 2
2, 8, 1
7 ,0 ,1
```

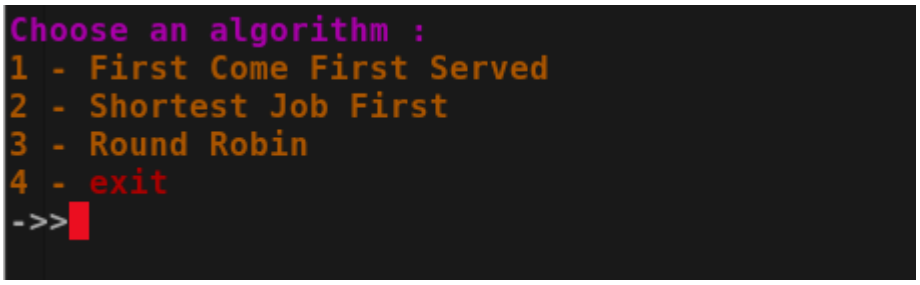
Pipe :(|)

Execution à partir d’un fichier :



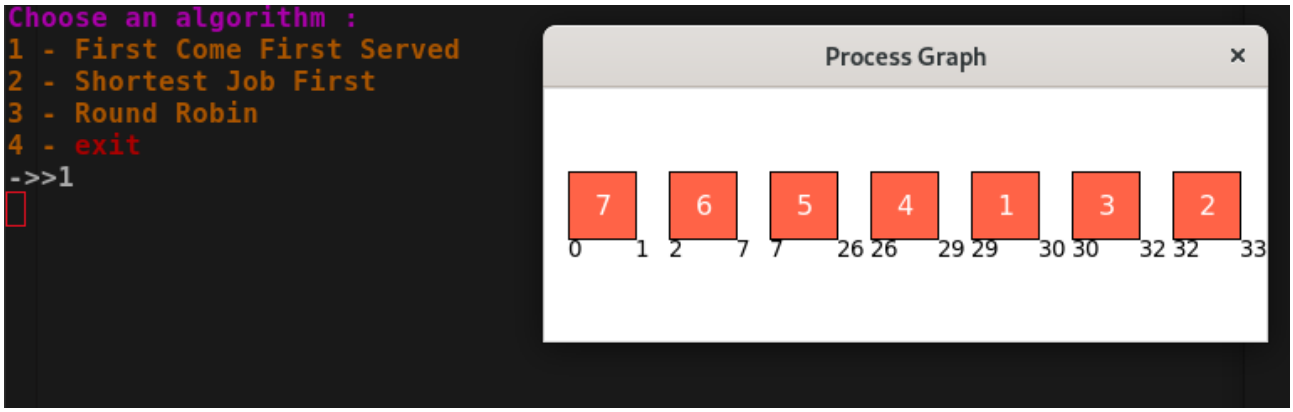
Partie II : Simulateur d’ordonnancement

Menu de l’ordonnanceur :

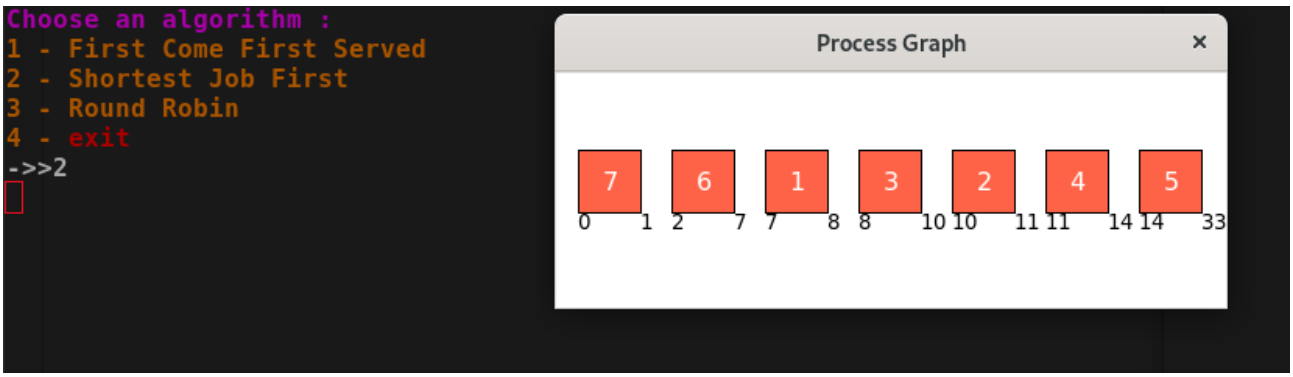


Exemple d’execution :

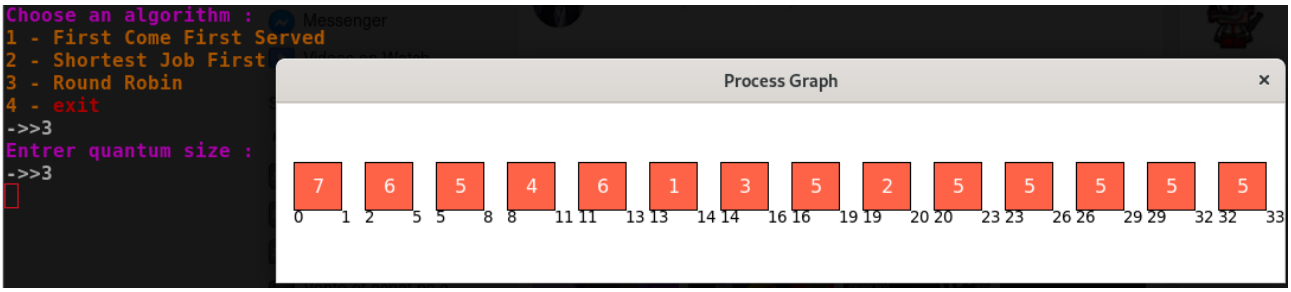
FIFO :



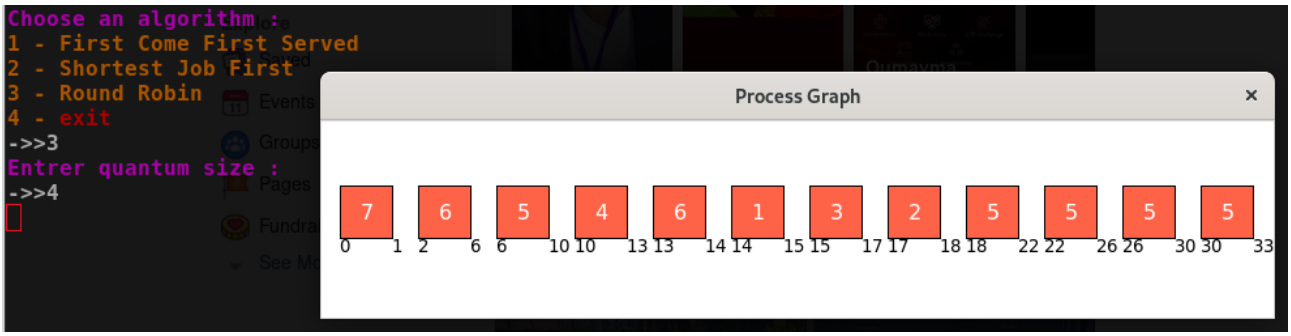
SJF :



Tourniquet : (quantum = 3)



Tourniquet : (quantum = 4)



ANNEXE I :

La fonction push() sépare les arguments de la commande et les enregistre dans un tableau.

```
start = Popen(args='commands/pwd2',stdout=PIPE,stderr=DEVNULL,encoding="utf-8").communicate()[0][:-1]

def push(data,command,Default=None):
    temp=["%s/commands/%s"%(start,command),]
    if not Default:
        for k in data.split(" ")[1:]:
            temp.append(k.replace(" ",""))
    return temp
```

La fonction call() execute les commandes en c et recupère leurs résultats.

```
def call(co,command,ret=None,Default=None):
    try:
        Default=open(Default,"r") if Default else None
        out=Popen(args=push(co,command,Default),stdout=PIPE,stderr=DEVNULL,stdin=Default,encoding="utf-8").communicate()[0][:-1]
        if not ret:
            print(colors.lightcyan+colors.bold+out+colors.reset)
        else :
            return(out)
    except:
        print(colors.lightred+colors.bold+"An Error Occured !"+colors.reset)
```

La fonction execute() sépare les commandes passées en une seule ligne et gére la redirection de l’entrée et sortie.

```
def execute(co,ret=None,Default=None):
    command=None
    eq=co.find("=")
    sp=co.find(" ")
    if "|" not in co:
        for j in aliases.keys():
            if j in co :
                command= (co[:len(co) if sp == -1 else sp] if eq == -1 else co[eq+1:]).strip().lower()
                co.replace(command,aliases[j])
                command =aliases[j]
                break
    if not command:
        command= (co[:len(co) if sp == -1 else sp] if eq == -1 else co[eq+1:]).strip().lower()
    for i in cmd :
        if '|' in co:
            try:
                out=None
                splt=co.split("|")
                for i in splt:
                    i.strip()
                    out=execute(i,True,"pipe_junk" if out else None)
                    if out:
                        with open("pipe_junk",'w') as f:
                            f.write(out)
                if out:
                    print(colors.lightcyan+colors.bold+out+colors.reset)
            except:
                print(colors.lightred+colors.bold+"An Error Occured !"+colors.reset)
            finally:
                break
        elif '>' in co:
            try:
                splt=co.split(">")
                with open(splt[1].replace(" ",""),'w') as f:
                    f.write(execute(splt[0].strip(),True)+"\n")
            except:
                print(colors.lightred+colors.bold+"An Error Occured !"+colors.reset)
            finally:
                break
        elif '<' in co:
            try:
                splt=co.split("<")
                execute(splt[0],None,splt[1].replace(" ",""))
            except:
                print(colors.lightred+colors.bold+"An Error Occured !"+colors.reset)
            finally:
                break
        elif i == command:
            if 'alias' in co:
                z = co.replace(" ","").split("=")
                aliases[z[0].replace("alias","")] = command
            elif 'help' == command:
                print("help not implemented")
            elif 'exit' == command:
                return 'exit'
            else:
                if 'cd2' == command:
                    try:
                        exec(call(co,command,True,None))
                    except:
                        print(colors.lightred+'No Such Directory : '+colors.yellow+colors.underline+co.replace(command,"").strip()+colors.reset)
                else:
                    return call(co,command,ret,Default)
            break
    else :
        print(colors.lightred+'Command '+colors.yellow+colors.underline+command+colors.reset+colors.lightred+' Not Found !! Check out ' +colors.BrightYellow+'HELP'+colors.lightred+' command'+colors.reset)
```

La fonction Shell() interprete les commandes et l’execution de plusieurs commandes d’un fichier texte.

```python
def shell():
    while 1 :
        pwd = Popen(args='%s/commands/pwd2'%start,stdout=PIPE,stderr=DEVNULL,encoding="utf-8").communicate()[0]
        machine = Popen(args=['uname -n'],shell=True,stdout=PIPE,stderr=DEVNULL,encoding="utf-8").communicate()[0][:-1]
        x = (colors.bold+colors.pink+os.environ['LOGNAME']+colors.BrightYellow+'@'+machine+':'+colors.BrightGreen+pwd+colors.lightgrey+'>>'+colors.BrightWhite).replace('\n', '')
        s = input(x).strip()
        if s == "":
            pass
        elif re.search("^./[a-zA-Z0-9]*$",s):
            s = s[2:]
            try:
                with open(s,'r') as f:
                    data=f.readline()[:-1].split(";")[:-1]
                    count = 0
                    while data:
                        for co in data:
                            count += 1
                            co=co.strip()
                            print(colors.pink+'\nCOMMAND '+str(count)+' : '+colors.BrightYellow+co+colors.reset+'\n')
                            execute(co)
                        data=f.readline()[:-1].split(";")[:-1]
            except:
                print(colors.lightred+'No Such File : '+colors.yellow+colors.underline+s+colors.reset)
        elif execute(s)=='exit':
            break
```

Les fonctions developpées en C :

Cat2.c :

```c
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
#include<fcntl.h>

int main(int argc,char *argv[])
{
    int fd, i, ch;
char* filename;
    if (argc==1){
scanf("%m[^¤]s", &filename);
printf("%s",filename);
/*fgets(filename, 1000000000, stdin);
printf("%s",filename);*/
}
else{
    for (i = 1; i < argc; i++) {
        fd = open(argv[i],O_RDONLY);

        if(fd < 0)
           printf("No such file %s\n",argv[i]);
        else{

        while(read(fd,&ch,1))

            write(STDOUT_FILENO,&ch,1);

        close(fd);
    }}
}
 }
```

ls2.c :

```c
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <dirent.h>

int main(int argc, char* argv[])
{
    DIR *mydir;
    struct dirent *myfile;
    struct stat mystat;
    if (argc < 2)argv[1]=".";
    mydir = opendir(argv[1]);
    while((myfile = readdir(mydir)) != NULL)
    {
        stat(myfile->d_name, &mystat);
        printf(" %s ", myfile->d_name);
    }
    printf("\n");
    closedir(mydir);
}
```

mkdir2.c :

```c
#include <stdio.h>

int main(int argc,char* argv[])
{
    int status;
    for(int i=1;i<argc;++i)
    {status= mkdir(argv[i]);
    if (status)
    printf("Unable to create directory %s\n",argv[i]);
    } }
```

pwd2.c :

```c
#include <unistd.h>
#include <stdio.h>

int main() {
    char cwd[1024];
    chdir("/path/to/change/directory/to");
    getcwd(cwd, sizeof(cwd));
    printf("%s\n", cwd);
}
```

rm2.c :

```
#include<stdio.h>

void main(int argc, char* argv[]){
int status;
for (int i=1;i<argc;i++){
status=remove(argv[i]);
if(status)
    printf("Failed to delete %s\n",argv[i]);
}}
```

touch2 :

```
#include <stdio.h>
int main(int argc,char* argv[])
{
FILE *fp;
for(int i=1;i<argc;i++){
if(fopen(argv[i], "r"))
{printf("File already exists %s\n" ,argv[i]);
}
else
fp  = fopen(argv[i], "w");
}}
```

ANNEXE II :

La fonction selectionSort() permet de trier un tableau de processus selon un critère spécifié (date d'arrivée , temps d'execution)

```
def selectionSort(alist,s):
    for i in range(len(alist)):
        minPosition = i
        for j in range(i+1, len(alist)):
            if alist[minPosition][s] > alist[j][s]:
                minPosition = j
        temp = alist[i]
        alist[i] = alist[minPosition]
        alist[minPosition] = temp
    return alist
```

La fonction scheduler() permet de planifier les processus selon l'option choisie (FIFO, SJF, Tourniquet).

```
def scheduler():
    inpt=input(colors.BrightMagenta+"Choose an algorithm :\n"+colors.BrightYellow+"1 - First Come First Served\n2 - Shortest Job First\n3 - Round Robin\n"+"4 - "+colors.BrightRed+"exit\n"+colors.BrightWhite+"->>")
    if inpt == "1":
        graph(data)
    elif inpt =="2":
        def get_sj(alist,index):
            temp=[alist[0],]
            for i in alist[1:]:
                if i[1] <= index:
                    temp.append(i)
                else:
                    break
            return selectionSort(temp,2)[0]
        index=data[0][1]
        temp=[]
        alist=data
        while len(alist) > 0 :
            k=get_sj(alist,index)
            temp.append(k)
            index+=k[2]
            alist.remove(k)
        graph(temp)
    elif inpt == "3":
        quantum=""
        while not quantum.isdigit():
            quantum=input(colors.BrightMagenta+"Entrer quantum size :\n"+colors.BrightWhite+"->>")
        quantum=int(quantum)
        save=data
        index = data[0][1]
        def get(alist,index):
            temp=[alist[0],]
            for i in alist[1:]:
                if i[1] <= index:
                    temp.append(i)
                else:
                    break
            return temp
        temp = get(save,index)
        ol=[]
        while len(save) > 0:
            print(temp)
            for i in temp:
                sub=min(quantum,i[2])
                ol.append((i[0],i[1],sub))
                i[2] -=sub
                index+=sub
                i[1]=index
                if i[2]<1:
                    save.remove(i)
            if save:
                index=max(index,save[0][1])
                temp = selectionSort(get(save,index),1)
        print("last",ol)
        graph(ol)
    elif inpt == "4":
        return
    else:
        print(colors.bold+colors.BrightYellow+"Wanna try again ?"+colors.reset)
    scheduler()
```

La fonction graph() permet de gérer l'interface graphique.

```
def graph(data):
    root = tk.Tk()
    root.title("Process Graph")
    c_width = len(data)*60+10
    c_height = 150
    c = tk.Canvas(root, width=c_width, height=c_height, bg='white')
    c.pack()
    x_stretch = 20
    x_width = 40
    x_gap = 15
    count=0
    for x,y in enumerate(data):
        x0 = x * x_stretch + x * x_width + x_gap
        x1 = x * x_stretch + x * x_width + x_width + x_gap
        s=max(count,y[1])
        c.create_rectangle(x0, 50, x1,90, fill="tomato",activefill="blue")
        c.create_text(x0 + 16, 80, anchor=tk.SW, text=y[0],activefill="black",fill="white",font="1")
        c.create_text(x0 , 105, anchor=tk.SW, text=str(s))
        count=s+y[2]
        c.create_text(x1, 105, anchor=tk.SW, text=str(count))
    root.mainloop()
```