

Concurso 2

Leonardo Moreira 71512

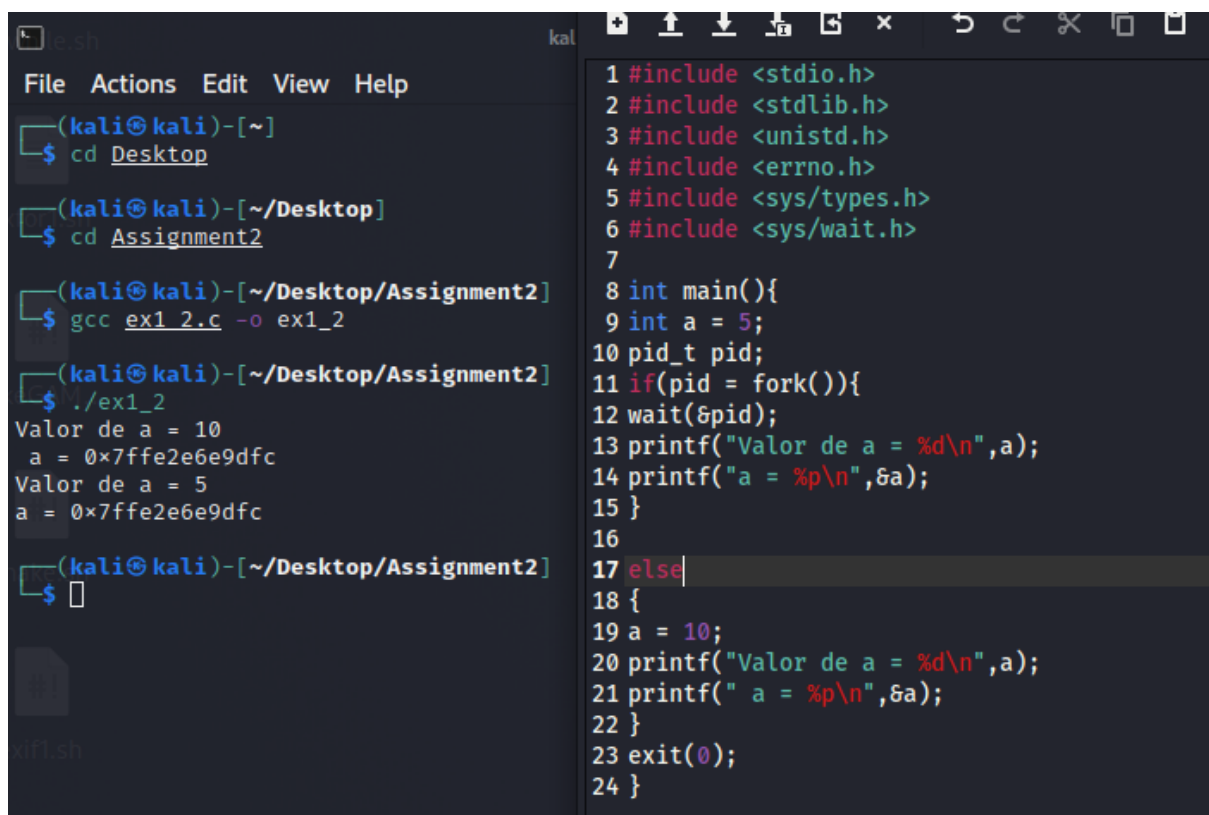
Exercício 1.

1.

São iniciados 4 processos pelo programa. Inicialmente temos o processo pai, depois são criadas 2 variáveis que correspondem a processos ID.

Depois, é criado o primeiro processo filho utilizando o primeiro fork, e o segundo fork irá repetir o processo do primeiro fork, por isso, irá originar mais 2 processos filhos onde um é originado do pai e outro do primeiro filho criado.

2.



```
e.sh
File Actions Edit View Help
(kali㉿kali)-[~]
$ cd Desktop
(kali㉿kali)-[~/Desktop]
$ cd Assignment2
(kali㉿kali)-[~/Desktop/Assignment2]
$ gcc ex1_2.c -o ex1_2
(kali㉿kali)-[~/Desktop/Assignment2]
$ ./ex1_2
Valor de a = 10
a = 0x7ffe2e6e9dfc
Valor de a = 5
a = 0x7ffe2e6e9dfc
(kali㉿kali)-[~/Desktop/Assignment2]
$
```

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <errno.h>
5 #include <sys/types.h>
6 #include <sys/wait.h>
7
8 int main(){
9     int a = 5;
10    pid_t pid;
11    if(pid = fork()){
12        wait(&pid);
13        printf("Valor de a = %d\n",a);
14        printf("a = %p\n",&a);
15    }
16
17    else
18    {
19        a = 10;
20        printf("Valor de a = %d\n",a);
21        printf("a = %p\n",&a);
22    }
23    exit(0);
24 }
```

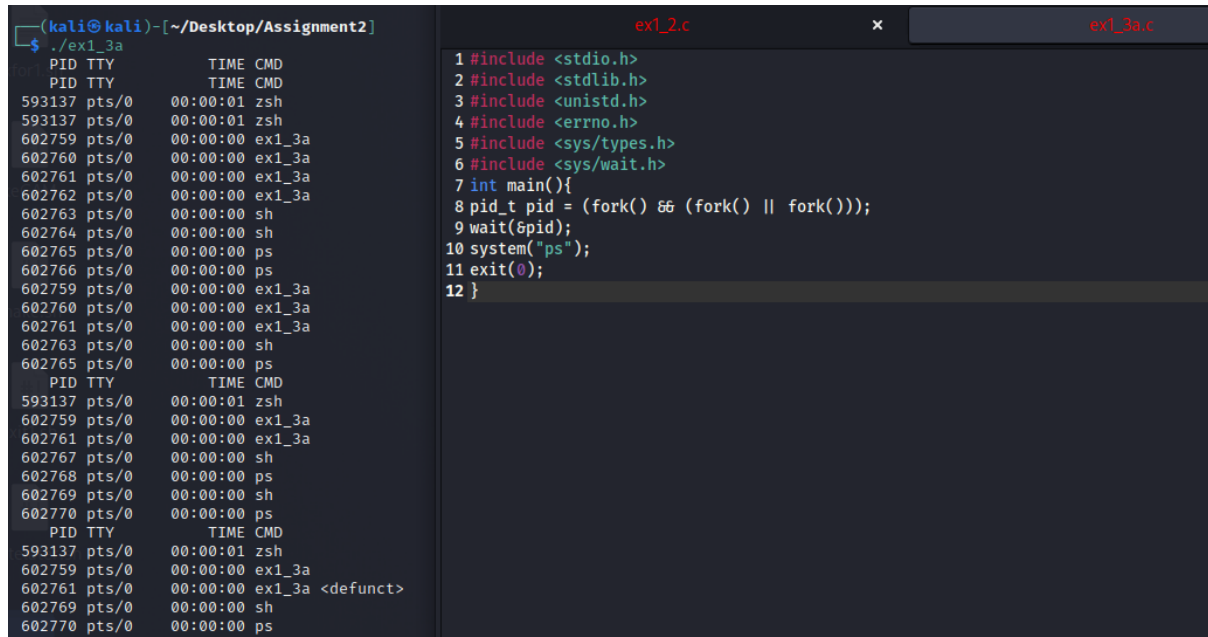
A variável `a` é instanciada com o valor 5, o pai cria um filho, depois o pai espera o filho acabar, o filho muda o valor de `a` para 10 depois dá print do valor de `a`(10) e do apontador e o filho fecha.

O pai depois dá print do valor de `a`(5) e do apontador que é igual ao do filho e depois o programa acaba.

O programa vai dar print do valor 5, 10 e dos respectivos apontadores porque os forks não partilham memória e fazem cópias idênticas. Apesar de mostrar o mesmo apontador o lugar de memória é diferente.

3.

a)



The screenshot shows a terminal window on the left and a code editor on the right. The terminal window title is "(kali@kali)-[~/Desktop/Assignment2]". It shows the command `./ex1_3a` being executed, followed by a series of process listings. The listings show various processes running, including `zsh`, `ex1_3a`, `sh`, and `ps`. The code editor window title is "ex1_2.c". It contains the following C code:

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <errno.h>
5 #include <sys/types.h>
6 #include <sys/wait.h>
7 int main(){
8     pid_t pid = (fork() && (fork() || fork()));
9     wait(&pid);
10    system("ps");
11    exit(0);
12 }
```

No caso de AND(&&), após avaliação do operando esquerdo, o operando direito será avaliado apenas se o operando esquerdo for diferente de zero.

Ou seja na expressão `(fork() && (fork() || fork()))`; o operando da direita `(fork() || fork())` apenas irá ser feito para o processo pai pois o operando esquerdo é diferente de zero

No caso de OR(||), após avaliação do operando esquerdo, o operando direito será avaliado apenas se o operando esquerdo for zero.

Ou seja na expressão `fork() || fork()` o operando da direita vai ser para o processo filho porque o da esquerda é igual a zero.

Ou seja se tivermos a falar de processos filhos e pai temos que o pai vai ter dois filhos (p1 e p2 por exemplo), o filho p2 vai ter um filho (p3) e todos eles correm o programa "ps".

Como todos executam o programa ao mesmo tempo, o output vem intercalado.

b)

```
(kali@kali) - [~/Desktop/Assignment2]
$ ./ex1_3b
PID TTY          TIME CMD
593137 pts/0        00:00:03 zsh
644333 pts/0        00:00:00 ex1_3b
644334 pts/0        00:00:00 ex1_3b
644335 pts/0        00:00:00 ex1_3b
644336 pts/0        00:00:00 ex1_3b
644337 pts/0        00:00:00 sh
644338 pts/0        00:00:00 ps
PID TTY          TIME CMD
593137 pts/0        00:00:03 zsh
644333 pts/0        00:00:00 ex1_3b
644335 pts/0        00:00:00 ex1_3b
644336 pts/0        00:00:00 ex1_3b
644339 pts/0        00:00:00 sh
644340 pts/0        00:00:00 ps
PID TTY          TIME CMD
593137 pts/0        00:00:03 zsh
644333 pts/0        00:00:00 ex1_3b
644335 pts/0        00:00:00 ex1_3b
644341 pts/0        00:00:00 sh
644342 pts/0        00:00:00 ps
PID TTY          TIME CMD
593137 pts/0        00:00:03 zsh
644333 pts/0        00:00:00 ex1_3b
644343 pts/0        00:00:00 sh
644344 pts/0        00:00:00 ps

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <errno.h>
5 #include <sys/types.h>
6 #include <sys/wait.h>
7 int main(){
8     pid_t child1 = fork();
9     pid_t child2, child3;
10    if(child1 && ( (child2 = fork() ) || (child3 = fork() ) ) ){ //pai, filho2
11        if(child2){
12            wait(&child1);
13            wait(&child2);
14        }else{ //é o filho p2
15            wait(&child3);
16        }
17        system("ps");
18    }
19    else{ //filho1 e filho3
20        if(child1) // é o filho3
21            usleep(100000);
22        system("ps");
23    }
24    exit(0);
25 }
```

Exercício 2.

1.

```
(kali@kali) - [~/Desktop/Assignment2]
$ gcc ex2_1.c -o ex2_1

(kali@kali) - [~/Desktop/Assignment2]
$ ./ex2_1
Eu sou o pai, minha identificação é <623554>
Eu sou o pai, minha identificação é <623554>
Eu sou o pai, minha identificação é <623554>
Eu sou o filho, meu pai é <623554>
Eu sou o filho, meu pai é <623554>
Eu sou o filho, meu pai é <623554>
Eu sou o filho, meu pai é <623554>
Eu sou o filho, meu pai é <623554>
Eu sou o filho, meu pai é <623554>

(kali@kali) - [~/Desktop/Assignment2]
$

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <errno.h>
5 #include <sys/types.h>
6 #include <sys/wait.h>
7
8 int main(){
9
10    pid_t child = fork();
11    if(child == 0){
12        usleep(10000);
13        for(int i = 0; i < 5; i++)
14            printf("Eu sou o filho, meu pai é <623554>\n", getppid());
15    }
16    else{
17        for(int i = 0; i < 3; i++)
18            printf("Eu sou o pai, minha identificação é <623554>\n", getpid());
19        sleep(1);
20    }
21    exit(0);
22 }
```

[illegible]

1.

```
(kali㉿kali)-[~/Desktop]
└─$ who; ps; ls
kali      tty7      2022-03-17 12:53 (:0)
          PID TTY          TIME CMD
485271 pts/0    00:00:00 zsh
506791 pts/0    00:00:00 ps
Assignment2 examadmission.sh exwhile.sh kali-linux-wallpaper-v7.png snake.sh
cmpnames.sh exfor1.sh file1.txt lab1 test
deletelater.sh exif1.sh file2.txt snakeGAME.sh test2

(kali㉿kali)-[~/Desktop]
└─$ who & ps & ls -l
[1] 506832
[2] 506833
kali      tty7      2022-03-17 12:53 (:0)
[1] - done      who
total 1704
-rwxr--r-- 1 kali kali      156 Apr 15 18:37 Assignment2
-rwxr--r-- 1 kali kali      110 Mar  8 09:43 cmpnames.sh
-rwxr--r-- 1 kali kali       60 Mar  8 13:17 deletelater.sh
-rwxr--r-- 1 kali kali       72 Mar  8 10:24 examadmission.sh
-rwxr--r-- 1 kali kali       0 Mar  8 10:06 exfor1.sh
-rwxr--r-- 1 kali kali       86 Mar  9 18:18 exif1.sh
-rwxr--r-- 1 kali kali      134 Mar  8 09:51 exwhile.sh
-rwxr--r-- 1 kali kali       9 Mar  8 05:48 file1.txt
-rwxr--r-- 1 kali kali       18 Mar  8 05:55 file2.txt
-rwxr--r-- 1 kali kali 1683326 Mar 22 22:38 kali-linux-wallpaper-v7.png
drwxr-xr-x 3 kali kali     4096 Mar 22 22:25 lab1
-rwxr--r-- 1 kali kali     5249 Mar  8 10:20 snakeGAME.sh
-rwxr--r-- 1 kali kali     5249 Mar  8 10:24 snake.sh
-rwxr--r-- 1 kali kali       59 Mar  8 10:24 test
drwxr-xr-x 2 kali kali     4096 Mar  8 04:39 test2

          PID TTY          TIME CMD
485271 pts/0    00:00:01 zsh
506833 pts/0    00:00:00 ps
[2] + done      ps
(kali㉿kali)-[~/Desktop]
└─$
```

No i) acontece tudo ao mesmo tempo, enquanto que no ii) cada processo espera acabar para começar outro logo a seguir.

2.

```
(kali@kali)-[~/Desktop/Assignment2]
$ ./ex3_2
kali      tty7      2022-03-17 12:53 (:0)
total 88
-rwxr--r-- 1 kali kali    330 Apr 17 14:54 ex1_2.c
-rwxr-xr-x 1 kali kali  16272 Apr 17 16:37 ex1_3a
-rwxr--r-- 1 kali kali    216 Apr 17 16:38 ex1_3a.c
-rwxr-xr-x 1 kali kali  16224 Apr 17 16:45 ex1_3b
-rwxr--r-- 1 kali kali    205 Apr 17 16:44 ex1_3b.c
-rwxr-xr-x 1 kali kali  16272 Apr 17 16:52 ex2_1
-rwxr--r-- 1 kali kali    348 Apr 17 16:52 ex2_1.c
-rwxr--r-- 1 kali kali    599 Apr 16 07:58 ex2_question1.c
-rwxr-xr-x 1 kali kali  16272 Apr 17 17:01 ex3_2
-rw-r--r-- 1 kali kali    306 Apr 17 17:01 ex3_2.c
  PID TTY          TIME CMD
 593137 pts/0      00:00:02 zsh
 619921 pts/0      00:00:00 ex3_2
 619922 pts/0      00:00:00 ps
 619923 pts/0      00:00:00 who <defunct>

(kali@kali)-[~/Desktop/Assignment2]
$
```

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <errno.h>
5 #include <sys/types.h>
6 #include <sys/wait.h>
7
8 int main(){
9     if(fork()){
10        usleep(10000);
11    } else if(fork()){
12        execlp("ps", "ps", NULL);
13    }
14    else if(fork()){
15        execlp("who", "who", NULL);
16    }else{
17        execlp("ls", "ls", "-l", NULL);
18    }
19    exit(0);
20 }
```

3.

```
(kali@kali)-[~/Desktop/Assignment2]
$ gcc ex3_3.c -o ex3_3

(kali@kali)-[~/Desktop/Assignment2]
$ ./ex3_3
kali      tty7      2022-03-17 12:53 (:0)
  PID TTY          TIME CMD
 593137 pts/0      00:00:02 zsh
 621612 pts/0      00:00:00 ex3_3
 621613 pts/0      00:00:00 ps
total 108
-rwxr--r-- 1 kali kali    330 Apr 17 14:54 ex1_2.c
-rwxr-xr-x 1 kali kali  16272 Apr 17 16:37 ex1_3a
-rwxr--r-- 1 kali kali    216 Apr 17 16:38 ex1_3a.c
-rwxr-xr-x 1 kali kali  16224 Apr 17 16:45 ex1_3b
-rwxr--r-- 1 kali kali    205 Apr 17 16:44 ex1_3b.c
-rwxr-xr-x 1 kali kali  16272 Apr 17 16:52 ex2_1
-rwxr--r-- 1 kali kali    348 Apr 17 16:52 ex2_1.c
-rwxr--r-- 1 kali kali    599 Apr 16 07:58 ex2_question1.c
-rwxr-xr-x 1 kali kali  16272 Apr 17 17:01 ex3_2
-rw-r--r-- 1 kali kali    306 Apr 17 17:01 ex3_2.c
-rwxr-xr-x 1 kali kali  16272 Apr 17 17:10 ex3_3
-rw-r--r-- 1 kali kali    335 Apr 17 17:09 ex3_3.c

(kali@kali)-[~/Desktop/Assignment2]
$
```

```
ex1_2.c x  ex1_3a.c x  ex1_3b.c
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <errno.h>
5 #include <sys/types.h>
6 #include <sys/wait.h>
7
8 int main(){
9     pid_t child;
10
11     if((child = fork())){
12         wait(&child);
13         execlp("ls", "ls", "-l", NULL);
14     }
15     else if((child = fork())){
16         wait(&child);
17         execlp("ps", "ps", NULL);
18     }
19     else{
20         execlp("who", "who", NULL);
21     }
22     exit(0);
23 }
```