Lab02a-Processos

Nome:Hao Yue Zheng-10408948
Nome:Smuel Zheng-10395781
Nome:Vitor Pasquarelli Cinalli-10401806

Linker de README
https://github.com/mauriciohao/Sistema-operacional.git

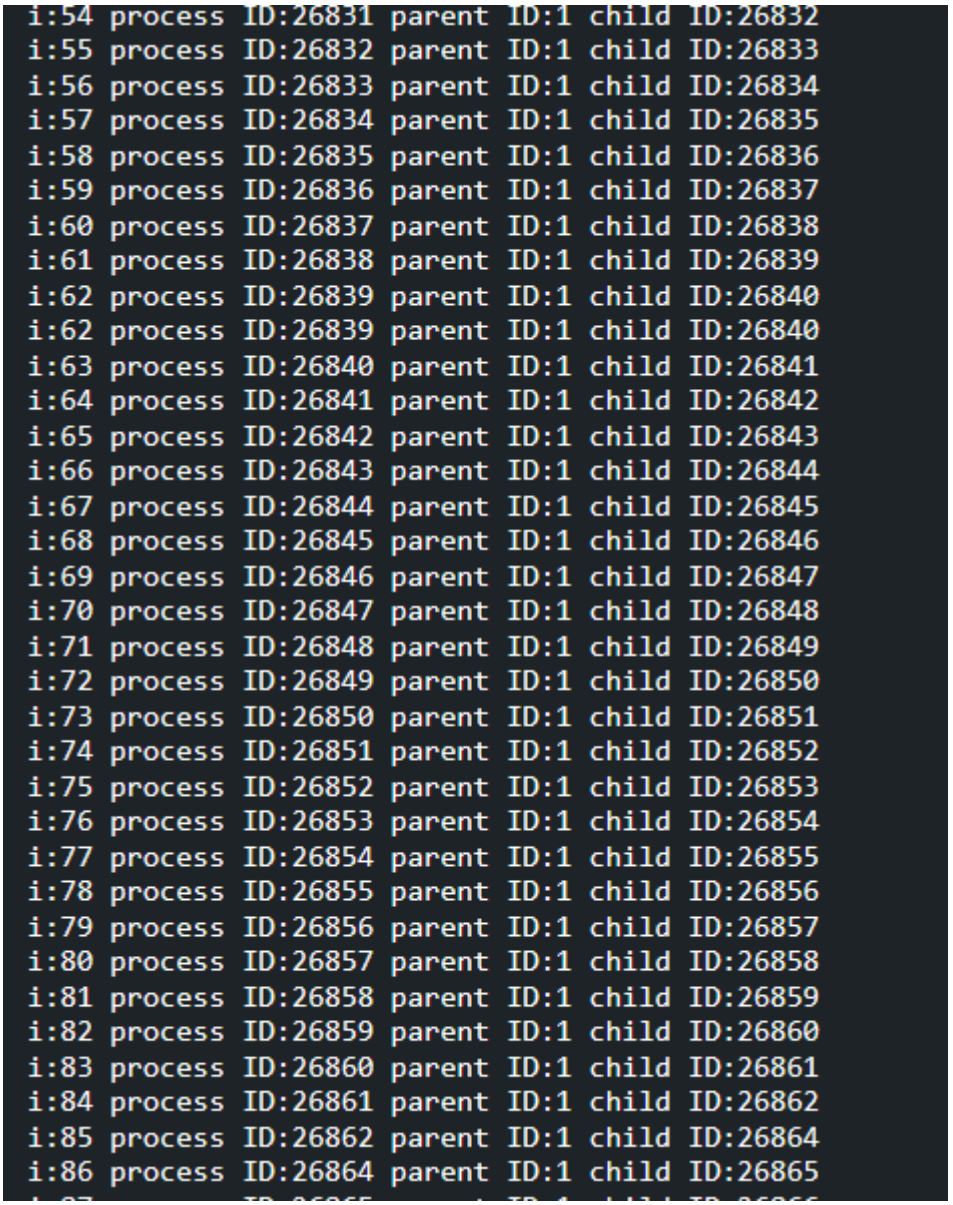
1- Rode o programa anterior para valores grandes de n. As mensagens sempre estarão ordenadas pelo valor de i?
Codigo

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int main (int argc, char *argv[]) {
   pid_t childpid = 0;

   int i, n;
   if (argc != 2){ /* check for valid number of command-line arguments */
     fprintf(stderr, "Usage: %s processes\n", argv[0]);
     return 1;
   }
   n = atoi(argv[1]);
   for (i = 1; i < n; i++)
       if (childpid = fork())
           break;
   fprintf(stderr, "i:%d process ID:%ld parent ID:%ld child ID:%ld\n",i, (long)getpid(),
(long)getppid(), (long)childpid);
   return 0;
}
```

Resolução

As saídas estão ordenadas pelo valor de ‘i’ porque a função ‘fprintf’ é chamada dentro do ‘loop’ for que está iterando em ‘i’ de ‘1’ até ‘n-1’.



2-O que acontece se o programa anterior escreve-se as mensagens para sys.stdout, usando print, ao invés de para sys.stderr?
Codigo

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int main (int argc, char *argv[]) {
   pid_t childpid = 0;
   int i, n;
   if (argc != 2){ /* check for valid number of command-line arguments */
```

```
      fprintf(stdout, "Usage: %s processes\n", argv[0]);
      return 1;
   }
   n = atoi(argv[1]);
   for (i = 1; i < n; i++)
       if (childpid = fork())
          break;
   fprintf(stdout, "i:%d process ID:%ld parent ID:%ld child ID:%ld\n",i, (long)getpid(),
(long)getppid(), (long)childpid);
   return 0;
}
```

Resolução

Se o programa anterior escrever as mensagens para sys.stdout usando printf em vez de fprintf para sys.stderr, as mensagens serão enviadas para a saída padrão em vez da saída de erro padrão.

```
i:10 process ID:29085 parent ID:1 child ID:0
^C
[ec2-user@ip-172-31-91-212 ~]$ ./run 100
i:1 process ID:29086 parent ID:2250 child ID:29087
[ec2-user@ip-172-31-91-212 ~]$ i:2 process ID:29087 parent ID:1 child ID:29088
i:3 process ID:29088 parent ID:1 child ID:29090
i:4 process ID:29090 parent ID:1 child ID:29091
i:5 process ID:29091 parent ID:1 child ID:29092
i:6 process ID:29092 parent ID:1 child ID:29093
i:7 process ID:29093 parent ID:1 child ID:29094
i:8 process ID:29094 parent ID:1 child ID:29095
i:9 process ID:29095 parent ID:1 child ID:29096
i:10 process ID:29096 parent ID:1 child ID:29097
i:11 process ID:29097 parent ID:1 child ID:29098
i:12 process ID:29098 parent ID:1 child ID:29099
i:13 process ID:29099 parent ID:1 child ID:29100
i:14 process ID:29100 parent ID:1 child ID:29101
i:15 process ID:29101 parent ID:1 child ID:29102
i:16 process ID:29102 parent ID:1 child ID:29103
i:17 process ID:29103 parent ID:1 child ID:29104
i:18 process ID:29104 parent ID:1 child ID:29105
i:19 process ID:29105 parent ID:1 child ID:29106
i:20 process ID:29106 parent ID:1 child ID:29107
i:21 process ID:29107 parent ID:1 child ID:29108
i:22 process ID:29108 parent ID:1 child ID:29109
i:23 process ID:29109 parent ID:1 child ID:29110
i:24 process ID:29110 parent ID:1 child ID:29111
i:25 process ID:29111 parent ID:1 child ID:29112
i:26 process ID:29112 parent ID:1 child ID:29113
i:27 process ID:29113 parent ID:1 child ID:29114
i:28 process ID:29114 parent ID:1 child ID:29115
i:29 process ID:29115 parent ID:1 child ID:29116
i:30 process ID:29116 parent ID:1 child ID:29117
i:31 process ID:29117 parent ID:1 child ID:29118
i:32 process ID:29118 parent ID:1 child ID:29119
i:33 process ID:29119 parent ID:1 child ID:29120
i:34 process ID:29120 parent ID:1 child ID:29121
i:35 process ID:29121 parent ID:1 child ID:29122
i:36 process ID:29122 parent ID:1 child ID:29123
```

```
i:56 process ID:29142 parent ID:1 child ID:29143
i:57 process ID:29143 parent ID:1 child ID:29144
i:58 process ID:29144 parent ID:1 child ID:29145
i:59 process ID:29145 parent ID:1 child ID:29146
i:60 process ID:29146 parent ID:1 child ID:29147
i:61 process ID:29147 parent ID:1 child ID:29148
i:62 process ID:29148 parent ID:1 child ID:29149
i:63 process ID:29149 parent ID:1 child ID:29150
i:64 process ID:29150 parent ID:1 child ID:29151
i:65 process ID:29151 parent ID:1 child ID:29152
i:66 process ID:29152 parent ID:1 child ID:29153
i:67 process ID:29153 parent ID:1 child ID:29154
i:68 process ID:29154 parent ID:1 child ID:29155
i:69 process ID:29155 parent ID:1 child ID:29156
i:70 process ID:29156 parent ID:1 child ID:29157
i:71 process ID:29157 parent ID:1 child ID:29158
i:72 process ID:29158 parent ID:1 child ID:29159
i:73 process ID:29159 parent ID:1 child ID:29160
i:74 process ID:29160 parent ID:1 child ID:29161
i:75 process ID:29161 parent ID:1 child ID:29162
i:76 process ID:29162 parent ID:1 child ID:29163
```

Exemplos - Processos

Identificação de Processos
Codigo

```
#include <stdio.h>
#include <unistd.h>
int main (void) {
   printf("I am process %ld\n", (long)getpid());
   printf("My parent is %ld\n", (long)getppid());
   return 0;
}
```

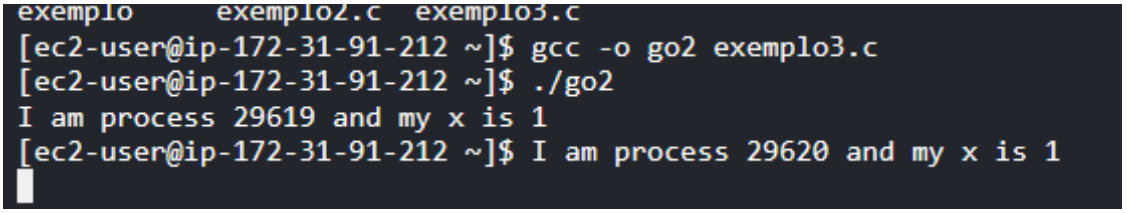
Resolução

```
[ec2-user@ip-172-31-91-212 ~]$ sudo nano
[ec2-user@ip-172-31-91-212 ~]$ gcc -o go1  exemplo2.c
[ec2-user@ip-172-31-91-212 ~]$ ./go1
I am process 29506
My parent is 2250
[ec2-user@ip-172-31-91-212 ~]$
```

Criação de Processos
Codigo

```c
#include <stdio.h>
#include <unistd.h>
  int main(void) {
  int x;
  x = 0;
  fork();
  x = 1;
  printf("I am process %ld and my x is %d\n", (long)getpid(), x);
  return 0;
}
```

Resolução



fork - pai e filho
Codigo

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
int main(void) {
   pid_t childpid;
   childpid = fork();
   if (childpid == -1) {
      perror("Failed to fork");
      return 1;
    }
    if (childpid == 0) /* child code */
        printf("I am child %ld\n", (long)getpid());
    else /* parent code */
       printf("I am parent %ld\n", (long)getpid());
    return 0;
}
```

Resolução

```
[ec2-user@ip-172-31-91-212 ~]$ sudo nano
[ec2-user@ip-172-31-91-212 ~]$ gcc -o go3 exemplo4.c
[ec2-user@ip-172-31-91-212 ~]$ ./go3
I am parent 29732
[ec2-user@ip-172-31-91-212 ~]$ I am child 29733
```

Pegando o Process ID
Codigo

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
int main(void) {
   pid_t childpid;
   pid_t mypid;
   mypid = getpid();
   childpid = fork();
   if (childpid == -1) {
      perror("Failed to fork");
      return 1;
    }
    if (childpid == 0) /* child code */
       printf("I am child %ld, ID = %ld\n", (long)getpid(), (long)mypid);
    else /* parent code */
       printf("I am parent %ld, ID = %ld\n", (long)getpid(), (long)mypid);
    return 0;
}
```

Resolução

```
[ec2-user@ip-172-31-91-212 ~]$ sudo nano
[ec2-user@ip-172-31-91-212 ~]$ gcc -o go5 exemplo5.c
[ec2-user@ip-172-31-91-212 ~]$ ./go5
I am parent 29838, ID = 29838
[ec2-user@ip-172-31-91-212 ~]$ I am child 29839, ID = 29838
```