Aluno: Victor Pessoa Oliveira Ortins
Matrícula: 20210024667

Github gist (Se quiser poder mexer melhor no código kkkk):
https://gist.github.com/VictorOrtins/9b34e1f921579cda0db90001f3710f52

```c
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>

#include "cores.c" // Vamos abstrair que aqui tem as cores kkkkk

int main(void) {

   printf("\n\n%s---- INÍCIO ----%s\n\n", AMARELO, RESET); // Início do programa

   // Definição das variáveis que armazenarão os pids dos processos
   // P1 terá os pids de F1 e F2. F1 e F2 terão o pid de P1, estando as variáveis de seus- respectivos pids zerados e
   // N1, N2, N3 e N4 terão os pids dos seus pais e de P1, estando as variáveis de seus respectivos pids zeradas
   int pid_p1 = -1, pid_f1 = -1, pid_f2 = -1, pid_n1 = -1, pid_n2 = -1, pid_n3 = -1, pid_n4 = -1;

   pid_p1 = (int) getpid(); // Pegando o pid de p1

   pid_f1 = (int) fork(); // Criando o processo F1. P1 recebe o pid de F1 e F1 recebe 0

   // Queremos, por enquanto, que apenas P1 tenha um filho. Então precisamos saber se o processo atual é P1
   // Comparamos se o pid atual é de fato o mesmo armazenado na variável P1 (que é compartilhada com todos os processos)
   if ((int) getpid() == pid_p1) {
      pid_f2 = (int) fork(); // Criando o processo F2
      // O processo P1 espera seus filhos terminarem de fazer o que precisam fazer
      // Pra forçar que esses waits sejam executados apenas em P1, deveria ter sido feito um if que nem o de fora desse bloco
      // Porém, como F2 apenas passa direto desses waits, já que não possui filhos, isso não é necessário
      wait(NULL);
      wait(NULL);
   }

   // Apenas no processo F1 pid_f1 está como 0. Em P1 ela está com o pid de F1
   if (pid_f1 == 0) {
      pid_n1 = (int) fork(); // Criamos o processo N1
      if (pid_n1 == 0) { // Se estamos no processo N1, já que apenas ela possui pid_n1 como 0
         execl("/bin/ls", "ls", NULL); // Executamos o comando de listar os arquivos e pastas no diretório
      }
      else { // Nesse caso, se não somos n1, obrigatoriamente somos f1, já que o if de pid_f1 == 0 foi feito acima
         pid_n2 = (int) fork(); // Criamos o processo N2
         if (pid_n2 == 0) { // Tu já entendeu o pq disso, deixe de onda
            execl("/bin/uname", "uname", "-a", NULL); // Executamos
         }
         else {
            wait(NULL); // Esperamos um dos processos filhos acabarem
            wait(NULL); // Esperamos o outro processo filho acabar
```

```
            sleep(1); // Sleep para que os Fs executem depois de todos os Ns
            // Print das informações de F1 e de P1
            printf("\n%s---- PROCESSO F1 de PID %d----%s\n%s Filho de P1 de PID %d%s\n\n\n",
VERMELHO, pid_p1, RESET, MAGENTA, (int)getpid(), RESET);
         }
      }
   }
   // Se formos o processo F2
   else if (pid_f2 == 0) {
      pid_n3 = (int) fork(); // Criamos N3
      if (pid_n3 == 0) { // Se formos n3
         execl("/bin/df", "df", "-h", NULL); // Executa o comando que indica o espaço de disco
restante e blocos disponíveis por dispositivo
      }
      else { // Se formos F2
         pid_n4 = (int) fork(); // Criamos N4
         if (pid_n4 == 0) { // Se formos N4
            execl("/bin/who", "who", NULL); // Executa o comando que indica os usuários ativos
         }
         else {
            wait(NULL); // Esperamos um dos processos filhos terminarem
            wait(NULL); // Esperampos o outro processo filho terminar
            sleep(1); // Sleep para que os Fs executem depois de todos os Ns
            // Print das informações de F2 e de P1
            printf("\n%s---- PROCESSO F2 de PID %d----%s\n%s Filho de P1 de PID %d%s\n\n\n",
VERMELHO, (int)getpid(), RESET, MAGENTA, pid_p1, RESET);
         }
      }
   }

   if ((int) getpid() == pid_p1) { // Se o processo for P1
      // Print das informações de P1
      printf("\n%s---- PROCESSO P1 de PID %d ----\n\n%s", VERDE, (int)getpid(), RESET);
   }

   return 0;
}
```

```
(base) victor@victor-IdeaPad-Gaming-3-15IHU6:~/Documentos/P6/Sistemas Operacionais/exec1$ gcc -o main main.c cores.c
(base) victor@victor-IdeaPad-Gaming-3-15IHU6:~/Documentos/P6/Sistemas Operacionais/exec1$ ./main


---- INICIO ----

Linux victor-IdeaPad-Gaming-3-15IHU6 6.5.0-17-generic #17~22.04.1-Ubuntu SMP PREEMPT_DYNAMIC Tue Jan 16 14:32:32 UTC 2 x86_64 x86_64 x86_64 G
NU/Linux
 cores.c                'Design sem nome.pdf'   Identificacao.odt   main    main2.c   main.pdf
'Design sem nome-1.pdf'   Identificacao-1.pdf    Identificacao.pdf   main2   main.c    RespostaParcial.pdf
victor   :1           2024-02-17 09:55 (:1)
Sist. Arq.      Tam. Usado Disp. Uso% Montado em
tmpfs           1,6G  2,5M  1,6G   1% /run
/dev/nvme0n1p6  165G   28G  129G  18% /
tmpfs           7,7G   39M  7,7G   1% /dev/shm
tmpfs           5,0M  4,0K  5,0M   1% /run/lock
efivarfs        184K  107K   73K  60% /sys/firmware/efi/efivars
/dev/nvme0n1p1  256M   77M  180M  30% /boot/efi
tmpfs           1,6G  120K  1,6G   1% /run/user/1000
/dev/nvme1n1p2  477G   92G  386G  20% /media/victor/ssd 512

---- PROCESSO F1 de PID 38177----
 Filho de P1 de PID 38178



---- PROCESSO F2 de PID 38179----
 Filho de P1 de PID 38177



---- PROCESSO P1 de PID 38177 ----
```

N4

N2

N1

N3