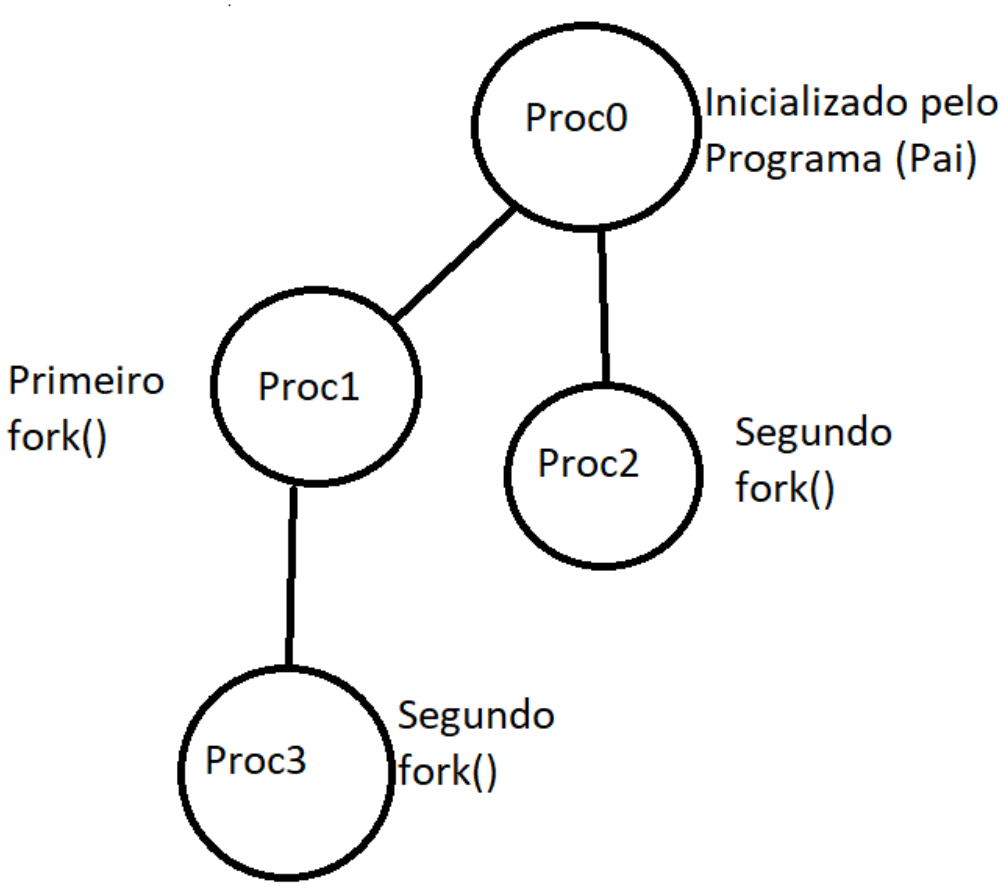
Concurso 2

EX I:

1. São iniciados 4 processos. O primeiro sendo inicializado pelo programa, o segundo pelo primeiro fork() e o terceiro e quarto processos são inicializados pelo ultimo fork().



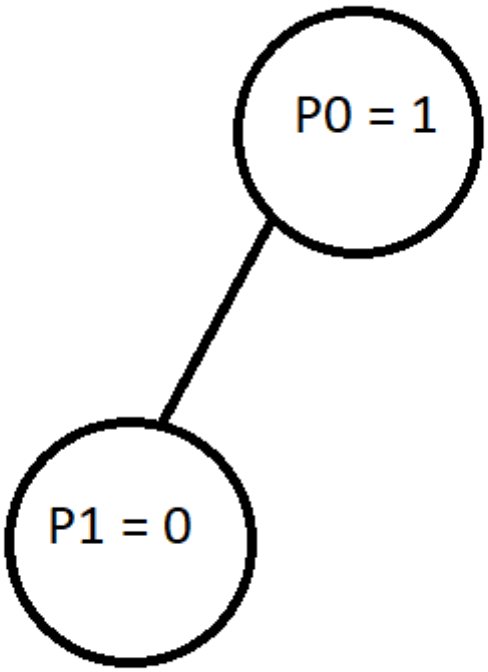
2.

```
int main(){int a = 5;
pid_t pid;
if(pid = fork())
 { wait(&pid);
printf("Valor de a = %d\n",a);
printf("a = %p\n",&a);
}
else
{
a = 10;
printf("Valor de a = %d\n",a);
printf(" a = %p\n",&a);
}
exit(0); }
```

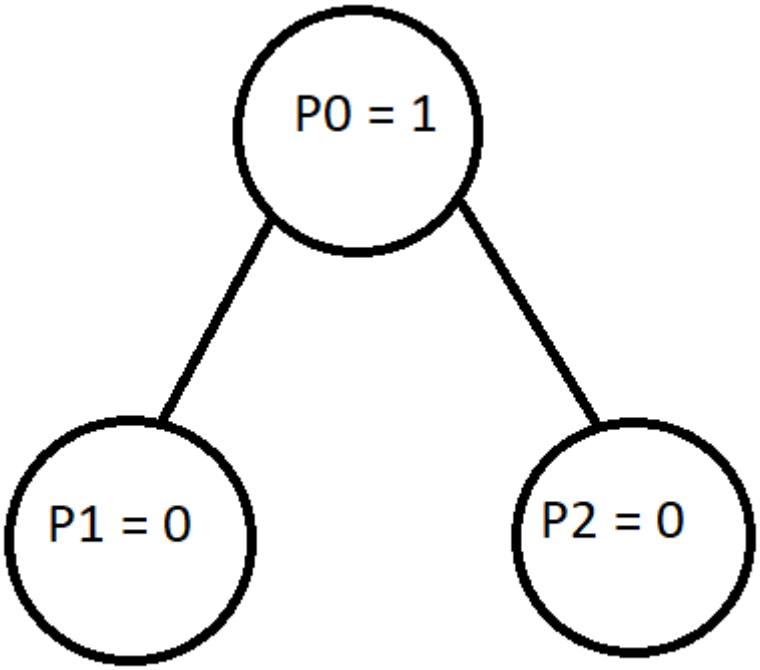
No primeiro if vamos criar um filho do processo onde este vai para o else, damos o valor de 10 a “a” e depois imprimimos. No output vemos primeiro o filho pois quando damos “pid = fork()” quando entramos no if é usado o wait(&pid) onde este vai esperar que o processo filho acabe antes de “executar” o pai.

3.

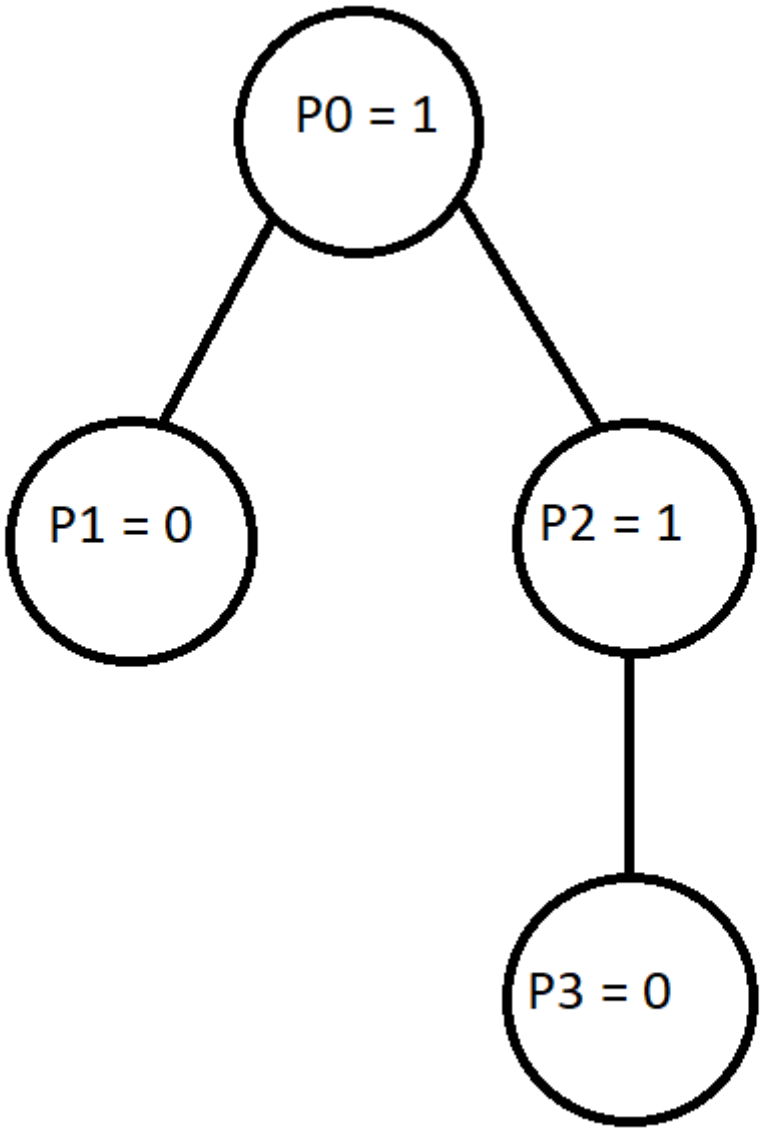
a)



Primeiro o programa executa um fork() onde o P1 é zero logo ele automaticamente sai da expressão, enquanto que o pai é 1 então continuamos na expressão



Em seguida é executado o primeiro fork dos “||” (fork() && (fork() || fork())), o P0 vai continuar com o valor de 1 então o processo não continua com o P0 e em vez disso P2 = 0 ou seja temos que continuar a expressão pois o futuro P3 pode ser igual a 1.



Por fim o processo P2 vira o pai do p3 pois agora vamos executar o ultimo fork() onde P2 previamente era zero mas agora é um mas como é o ultimo fork não interessa, é criado o ultimo processo P3 onde via ter o valor de 0.

b) O código consegue ser executado não é necessário fazer alterações.

EX II:

1)

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

int main(){
    pid_t pid;
    if(pid = fork()){
        wait(&pid);
        for(int i = 0; i < 3; i++)
            printf("Eu sou o pai, minha identificação é %d\n", getpid());
    }
    else{
        for(int i = 0; i < 5; i++)
            printf("Eu sou o filho, meu pai é %d\n", getppid());
    }
    exit(0);
}
```

2)

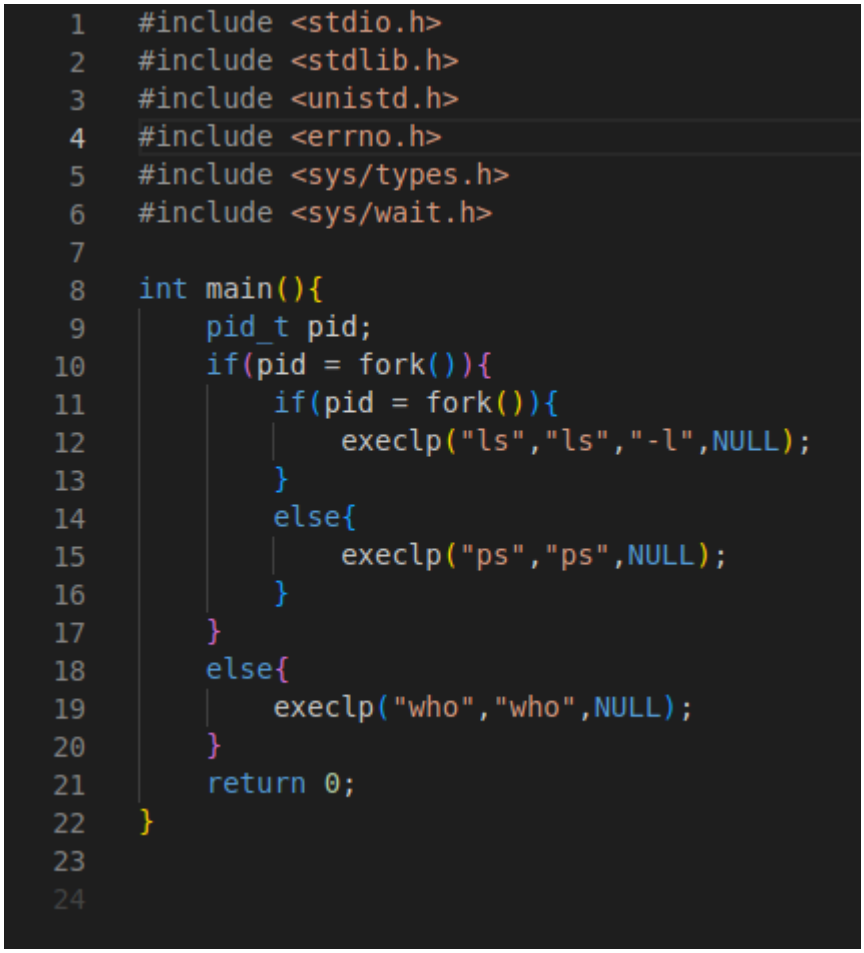
```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <sys/types.h>
#include <sys/wait.h>

int main(){
    pid_t process;
    pid_t process2;
    if(process = fork()){
        wait(&process);
        printf("Eu sou o pai, minha identificação é %d\n", getpid());
        for(int i = 0; i < 3; i++){
            if(fork() == 0){
                printf("Eu sou o filho, meu pai %d\n", getppid());
                exit(0);
            }
            else{
                wait(&process);
            }
        }
    }
    else{
        printf("Eu sou o filho, meu pai %d\n", getppid());
        for(int i = 0; i < 2; i++){
            if((process2 = fork()) == 0){
                printf("Eu sou o filho, meu pai %d\n", getppid());
                exit(0);
            }
            else{
                wait(&process2);
            }
        }

    }
    exit(0);
}
```

EX III:

1) A diferença entre i) e ii) é que no primeiro o comandos vao ser executados ao mesmo tempo enquanto que no ii) os comandos têm que esperar que os outros acabem, por exemplo para o

comando “ps” ser executado o comando “who” tem que acabar.

2)



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

int main(){
    pid_t pid;
    if(pid = fork()){
        if(pid = fork()){
            execlp("ls","ls","-l",NULL);
        }
        else{
            execlp("ps","ps",NULL);
        }
    }
    else{
        execlp("who","who",NULL);
    }
    return 0;
}
```

3)

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

int main(){
    pid_t pid;
    if(pid = fork()){
        wait(&pid);
        if(pid = fork()){
            wait(&pid);
            execlp("ls","ls","-l",NULL);
        }
        else{
            execlp("ps","ps",NULL);
        }
    }
    else{
        execlp("who","who",NULL);
    }
    return 0;
}
```