Stephanie Magalhães Fay
Pedro Paulo da Silva

1-

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

int main () {
   int pid = fork();

   if (pid != 0) {
         printf("PID pai: %d\n", getpid());
   }
   else {
         printf("PID filho: %d\n", getpid());
         exit(1);
   }
   return 0;
}
```

Ao realizar o fork o processo filho entra em espera com a função waitpid, que passa o controle para o processo pai, imprime seu pid e volta para o filho que encerra a execução com exit.

2-

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

int main () {
   int i;
   int contador = 1;
   int pid = fork();

   if (pid != 0) {
         printf("PID pai: %d\n", getpid());
         for(i = 0; i < 50; i++) {
               contador++;
               printf(" %d, ", contador);
         }
         printf("\n Esperando filho terminar\n");
   }
```

```
    else {

        printf("PID filho: %d\n", getpid());
        for(i = 0; i <= 100; i++) {
            contador += 2;
            printf(" %d, ", contador);
        }
        printf("\n Finalizando filho\n");
        exit(1);
    }
    return 0;
}
```

Não há concorrência porque o processo pai executa e então espera o filho executar e finalizar.

3-

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

int main () {
    int i;
    int contador = 1;
    int pid = fork();
    int status;

    if (pid != 0) {
        printf("PID pai: %d\n", getpid());
        for(i = 0; i < 50; i++) {
            contador++;
            printf(" %d, ", contador);
        }
        printf("\n Esperando filho terminar\n");
    }
    else {
        int pid2 = fork();
        int status2;

        if (pid2 != 0) {
            printf("PID filho: %d\n", getpid());
            for(i = 0; i < 100; i++) {
                contador += 2;
                printf(" %d, ", contador);
```

```
            }
            printf("\n Esperando neto terminar\n");
        }
        else {
            waitpid(pid, &status, 0);
            waitpid(pid2, &status2, 0);
            printf("PID neto: %d\n", getpid());
            for(i = 0; i < 150; i++) {
                contador += 3;
                printf(" %d, ", contador);
            }
            printf("\n Finalizando neto\n");
        }

        exit(1);
    }
    return 0;
}
```

Novamente não há concorrência pois o processo pai espera o filho que por sua vez espera o processo neto executar.

4-

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

int main () {
    int i;
    int contador = 1;
    int pid = fork();
    int status;

    if (pid != 0) {

        int pid2 = fork();
        int status2;

        if (pid2 != 0) {
            printf("PID pai: %d\n", getpid());
            for(i = 0; i < 50; i++) {
                contador++;
                printf(" %d, ", contador);
            }
```

```
                printf("\n Esperando filhos terminarem\n");
        }
        else {
                waitpid(pid, &status, 0);
                waitpid(pid2, &status2, 0);
                printf("PID filho 2: %d\n", getpid());
                for(i = 0; i <= 100; i++) {
                        contador += 2;
                        printf(" %d, ", contador);
                }
                printf("\n Finalizando filho 2\n");
                exit(1);
        }
    }
    else {
        printf("PID filho 1: %d\n", getpid());
        for(i = 0; i <= 100; i++) {
                contador += 2;
                printf(" %d, ", contador);
        }
        printf("\n Esperando filho 2 terminar\n");
    }
    return 0;
}
```

A diferença entre eles é que a espera entre os irmãos é feita no segundo filho e não no primeiro.