

Universidade Federal de Itajubá
SRSC02 – Sistemas Operacionais
2023005594 – Gabriel Toti
2017006513 – Diogo Nandes

Exercício 1:

O código foi compilado e executado de forma correta, sem a ocorrência de erros ou warnings.

Exercício 2:

Considerando que o `fork()` deve criar um processo para executar trechos de código distintos, o código não executou de maneira correta. O ideal seria que o segundo `printf()` aparecesse apenas uma vez, ao invés de duas, como ocorreu.

Exercício 5:

Em caso de sucesso, a função `fork()` pode retornar dois valores a depender do processo em execução no momento. No caso de o processo executado no momento ser o processo pai (que chamou o `fork` inicialmente), o valor retornado é o PID do processo filho criado. No caso de ser o processo filho, ele retornará 0.

Exercício 6:

O valor do PID do processo filho é atribuído pelo sistema operacional. Dessa forma, não é possível prever seu valor exato, já que ele depende do estado do sistema e dos processos que entram em execução. Entretanto, é possível dizer que um processo filho criado com o `fork` terá sempre um PID maior que o do processo pai.

Exercício 7:

A função utilizada para se obter o PID do processo executado atualmente é a `getpid()`.

Exercício 8:

O código apresentado cria 3 processos. Inicialmente, o processo principal P cria no primeiro `fork` o processo filho P1, que começa a executar o código apartir do segundo `fork`. Assim, P1 irá executar o segundo `fork`, criando o processo P2. Entretanto P segue sua execução e também chama o segundo `fork`, gerando mais um processo, o P3, resultando em 3 processos criados.

Exercício 9:

Apesar de dividirem a mesma memória virtual, o processo pai e o processo filho estão em partes diferentes da memória. Dessa forma, ao se alterar uma variável em qualquer um deles o valor se mantém o mesmo no outro processo, ou seja, se uma variável é declarada e em seguida modificada por um processo filho P1, seu valor é mantido o mesmo no processo pai P e nos outros processos filhos de P.

Exercício 10:

Resposta no arquivo ex10.c.

Exercício 11:

A função `wait()` irá retornar, em caso de sucesso, o valor do PID do primeiro processo filho do processo atualmente em execução a finalizar. Já em caso de erro, irá retornar -1.

Exercício 12:

Não, pois a função `wait` faz com que o processo atualmente em execução espere pela finalização de seu processo filho e não verifica a execução de seu processo pai.

Exercício 14:

As funções da família `exec` só irão retornar um valor inteiro (-1) em caso de erro, já que em caso de sucesso o processo passado como parâmetro da função irá substituir a execução do atual.

Exercício 15:

Não. Em caso de um processo que seja especificado e não encontre o processo a ser executado, por exemplo, o `exec` não será executado. Existem também outros fatores que podem impedir a execução como permissão para execução ou falta de memória por exemplo.

Exercício 16:

A função `system` também executa outros processos assim como a `exec`, porém ela não substitui o programa executado atualmente pelo programa solicitado como a `exec` faz.