Universidade Federal de Itajubá

SRSC02 – Sistemas Operacionais

Luara Perilli 202200841

Thiago Luiz de Matos 2024016073

Exercício Prático 01 - EP01 Processos

Exercício 1. O Código 1 executou corretamente?

R: Sim, o código 1 executou corretamente. Ele cria um novo processo com o comando fork(). O código continua sendo executado tanto pelo processo pai quanto pelo processo filho, resultando na impressão das mensagens "fim do código" duas vezes. O resultado esperado seria:

```
inicio do codigo
fim do codigo
fim do codigo
```

Exercício 2. Comente o resultado da execução do Código 1.

R: Após a chamada de fork(), dois processos executam de forma concorrente: o processo pai e o processo filho. Ambos executam o mesmo código a partir do ponto da chamada fork(), o que faz com que as linhas printf(“inicio do codigo”\n) e printf(“fim do codigo\n”) sejam executadas duas vezes, uma vez pelo pai e outra pelo filho.

Exercício 3. Escreva um programa usando a system call fork() para criar dois filhos do mesmo processo, isto é, o Pai P possui P1 e P2 como processos filhos.

R: Resposta dentro da pasta zip.

Exercício 4. Escreva um programa usando a system call fork() para criar uma hierarquia de 3 processos, isto é, P2 é filho de P1, e P1 é filho de P.

R: Resposta dentro da pasta zip.

Exercício 5. Responda: o que a system call fork() retorna em caso de sucesso?

R: Em caso de sucesso, o fork() retorna o **PID** (identificador do processo) do processo filho para o processo pai. No processo filho, fork() retorna **0**.

Exercício 6. Responda: qual o PID de um processo filho?

R: O PID do processo filho é um valor único que é atribuído pelo sistema operacional (obtido por meio da função getpid()), sendo inicialmente 0 quando o processo é criado.

Exercício 7. Responda: qual a função usada para se obter o PID de um processo?

R: A função usada para obter o PID de um processo é getpid().

Exercício 8. No total, quantos processos são criados no código abaixo?

R: No total, são criados **4 processos**. O primeiro fork() cria um processo filho, totalizando 2 processos. O segundo fork() é executado por ambos (pai e filho), gerando mais dois processos, totalizando 4.

Exercício 9. Descreva o que acontece se os processos pai e filho alteram o valor de uma variável declarada no código.

R: O processo pai e o processo filho têm o mesmo espaço de endereçamento virtual, o que significa que as variáveis podem parecer estar no mesmo endereço. No entanto, qualquer alteração em uma variável no processo pai ou no processo filho não afeta o outro processo, pois, após o fork(), ambos terão seu próprio espaço de memória física para cada alteração feita.

Exercício 10. Como verificar, em código, se o pai verdadeiro de um processo já se encerrou ou não?

R: Através da função getppid(). Se o valor retornado for **1**, significa que o processo pai original terminou e o processo filho foi adotado pelo processo init (PID 1).

Exercício 11. Responda: o que a system call wait() retorna em caso de sucesso? E de falha?

R: Em caso de sucesso, wait() retorna o PID do filho que terminou. Em caso de falha, retorna -1 e a variável erro é ajustada para indicar o erro.

Exercício 12. Responda: é possível usar wait() para fazer com que o processo filho espere o pai finalizar? Por quê?

R: Não, o wait() é usado pelo processo pai para esperar a finalização dos filhos, mas não pode ser usado para o filho esperar o pai. Os filhos não podem monitorar a finalização do pai diretamente.

Exercício 13. Escreva um programa que crie dois processos filhos do mesmo pai. O pai deve esperar ambos os filhos finalizarem antes de encerrar.

R: Resposta dentro da pasta zip.

Exercício 14: Quando a família de funções exec() retorna um valor inteiro?

R: As funções da família exec() retornam um valor inteiro apenas em caso de falha. Se a função for bem-sucedida, o processo original é substituído pelo novo processo e, portanto, não há retorno.

Exercício 15: O comando passado para a função exec() sempre executa. A afirmação é verdadeira ou falsa? Por quê?

R: Falsa. O comando passado para exec() pode falhar se o programa ou arquivo especificado não for encontrado ou se o processo não tiver permissões adequadas. Nesses casos, a função retorna **-1**.

Exercício 16: No Código 4, você usou a função system() para fazer com que outro processo fosse executado. Nos Códigos 5 e 6, você usou a função exec(). Explique como essas duas funções se diferenciam.

R: A função system() cria um novo processo filho para executar um comando de shell, enquanto a função exec() não cria um novo processo, apenas substitui o contexto do processo atual pelo novo processo, sem criar um novo processo filho.