# Universidade Federal de Itajubá

# SRSC02 – Sistemas Operacionais

Luara Perilli 202200841

Beatriz Guimarães do Nascimento 2023007113

## Exercício Prático 01 - EP01 Processos

**Exercício 1: Responda:** o que a função clone() retorna?

R: A função clone() retorna o PID (Process ID) do processo (thread) filho criado em caso de sucesso. Se a criação da thread falhar, ela retorna −1. Isso é semelhante ao que ocorre com a função fork(), mas com mais flexibilidade.

**Exercício 2: Responda:** qual a diferença entre processos criados usando fork() e usando clone()?

R: A principal diferença entre fork() e clone() reside no modo como os recursos são compartilhados, ou seja:

- O **fork()** cria uma cópia quase exata do processo pai, resultando em um novo processo com seu próprio espaço de memória, arquivos abertos, e recursos. Ainda, os processos criados por fork() não compartilham recursos com o processo pai, a não ser que sejam explicitamente compartilhados (por exemplo, usando memória compartilhada). Cada processo vai rodar de forma independente a partir do ponto onde o fork() foi chamado.
- O **clone()** permite maior controle sobre o compartilhamento de recursos. Por meio de flags, é possível especificar quais recursos (memória, sinalização, arquivos abertos, etc.) serão compartilhados entre o processo pai e o novo processo (ou thread). Assim, se CLONE_VM for especificado, o novo processo compartilhará o espaço de memória com o pai, agindo como uma thread.

**Exercício 3: Programe e responda:** o que acontece se a função principal encerrar sua execução, mas as threads por ela criadas não tiverem encerrado ainda?

R: Se a função principal encerrar sua execução enquanto as threads criadas ainda estão ativas, o comportamento pode variar dependendo da implementação e do sistema operacional, mas o mais comum é que todas as threads também sejam encerradas automaticamente. Portanto, é fundamental que a thread principal use phthread_join() para garantir que todas as threads filhas sejam esperadas e terminadas corretamente antes de finalizar o programa, evitando comportamentos inesperados e possíveis problemas de gerenciamento de processos.

**Exercício 4: Programe e responda:** o que acontece se uma thread invoca a função fork() durante sua execução?
R: Quando uma thread invoca fork(), o novo processo criado pelo fork() terá uma cópia do processo pai, incluindo todas as threads do processo pai no momento da chamada. Entretanto, tanto o processo pai quanto o filho continuarão a execução normalmente a partir do ponto onde o fork() foi chamado.

**Exercício 5: Programe e responda:** o que acontece se uma thread invoca a função clone() durante sua execução?
R: Se uma thread invoca clone() durante sua execução uma nova thread será criada, e ela pode compartilhar recursos com a thread que fez a chamada, dependendo das flags passadas para clone(). Por exemplo, se as flags incluírem CLONE_VM, a nova thread compartilhará o mesmo espaço de memória, permitindo comunicação e compartilhamento de dados entre ambas. A nova thread começará sua execução na função especificada na chamada de clone(), recebendo como argumento um ponteiro para a função que deve ser executada. Importante ressaltar que a thread original (pai) e a nova thread (filha) têm suas próprias pilhas, a menos que a flag CLONE_STACK seja utilizada para compartilhar a pilha. Além disso, ao utilizar clone(), o controle sobre a nova thread é importante, já que o programador deve gerenciar corretamente sua vida útil e recursos associados.

**Exercício 6: Programe e responda:** o que acontece se uma thread invoca uma das funções da família exec() durante sua execução?
R: Quando uma thread invoca uma função da família exec() (como execl(), execp(), etc.) o processo inteiro é substituído pelo novo programa que a função exec() carrega. Isso significa que, embora a thread que chamou exec() seja substituída, todas as threads do processo original são encerradas, pois o espaço de memória do

processo é totalmente substituído pelo novo programa. Assim, o resultado é que o programa atual é finalizado e o novo programa começa a execução. Importante notar que, como a função exec() não retorna em caso de sucesso, qualquer código após a chamada a exec() na thread que invocou a função não será executado, a menos que ocorra um erro durante a execução.
Além disso, se houver algum estado a ser mantido ou recursos a serem liberados, deve-se garantir que essas tarefas sejam concluídas antes da chamada exec(), já que, após essa chamada, o controle não retornará ao programa original.

**Exercício 7: Programe e responda:** o que acontece se a função principal encerrar sua execução, mas as threads por ela criadas com pthread_create() não tiverem encerrado ainda?
R: Quando a função principal (thread principal) termina sua execução, se houver outras threads ainda ativas, o comportamento pode ser o seguinte:

1. **Threads Ativas**: As threads criadas permanecerão ativas e continuarão executando até que terminem suas respectivas tarefas ou sejam explicitamente encerradas.
2. **Thread Principal**: Se a thread principal termina sem chamar pthread_exit(), o processo (e todas as threads associadas a ele) será encerrado de forma abrupta. O sistema operacional pode matar todas as threads em execução associadas a esse processo.
3. **Recursos**: As threads ativas podem não ter a chance de liberar recursos ou executar suas rotinas de limpeza.

**Exercício 8: Programe e responda:** o que acontece se uma thread criada com pthread_create() invoca a função fork() durante sua execução?
R: Se uma thread criada com pthread_create() invocar um fork() o comportamento do processo filho depende da forma com a qual o sistema lidará com a clonagem do processo. Após o fork(), no processo filho, somente a thread que executou o fork() será copiada para o novo processo, as outras threads do processo pai não existirão no processo filho. Entretanto, usar um fork() sem chamar exec() logo após pode levar a um deadlock, especialmente se a thread que chamou o fork() estava interagindo com recursos compartilhados.

**Exercício 9: Programe e responda:** o que acontece se uma thread criada com pthread_create() invoca a função clone() durante sua execução?
R: Se uma thread invoca a função clone() ela pode criar um novo processo ou thread com características compartilhadas ou isoladas, dependendo dos flags fornecidos a clone(). Isso dá mais controle do que o uso de fork() sozinho.

**Exercício 10: Programe e responda:** o que acontece se uma thread criada com pthread_create() invoca uma das funções da família exec() durante sua execução?
R: Se uma thread invoca uma função da família exec() o processo inteiro é substituído, o que encerra todas as threads em execução no processo. O processo continua com a nova imagem do programa, e a execução será retomada a partir do ponto de entrada do novo programa. Após a execução do exec() o controle não retorna ao código original, pois ele foi completamente substituído.

**Exercício 11:** Usando a API PThreads, crie um programa que leia um dois inteiros h e m representando hora e minuto e os repasse para uma outra thread. A thread deve imprimir “Bom dia”, “Boa tarde” ou “Boa noite”, de acorddo com o valor de hora e minuto recebidos.
R: Resposta no arquivo ex11.c

**Exercício 12:** Usando a API PThreads, crie um programa que leia um dois inteiros a e b do usuário. A seguir, crie duas funções filhas. A primeira função deve computar a soma dos valores pares no intervalo [a, b]. A segunda função deve computar o produto dos valores ímpares no intervalo [a,b]. As respostas devem ser impressas na tela pela função main().
R: Resposta no arquivo ex12.c

**Exercício 13:** Usando a API PThreads, crie um programa que declare um vetor de 20 inteiros aleatórios e o repasse para duas funções filhas. A primeira função deve computar a média dos elementos do vetor. A segunda função deve computar a mediana. As respostas devem ser impressas na tela pela função
main().
R: Resposta no arquivo ex13.c

**Exercício 14:** Usando a API PThreads, crie um programa que declare um vetor de 20 inteiros aleatórios e o repasse para duas funções filhas. A primeira função deve

computar a média dos elementos do vetor. A segunda função deve computar o desvio padrão. As respostas devem ser impressas na tela pela função main().

R: Resposta no arquivo ex14.c