Função Clone: Chamada de sistema usada para criar processos chamados de threads ou tarefas.

- Ela permite uma grande flexibilidade na forma como o novo processo compartilha recursos com o processo chamador (o processo pai).
- Ao chamar clone, você pode especificar um conjunto de sinalizadores que determinam exatamente quais recursos são compartilhados entre o processo pai e o filho. Nesse caso consistem em clone_fs, clone_files, clone_sighand, clone_vm
- A diferença para um fork() é que nessa função observamos um alto grau de controle sobre o que é compartilhado entre o processo pai e filho.

CLONE_FS

- O processo filho compartilha as mesmas informações do sistema de arquivos que o processo pai, isto é, diretório de trabalho atual e raiz do sistema de arquivos.

CLONE_FILES

- O processo filho compartilha as mesmas informações do sistema de arquivos que o processo pai, isto é, os mesmos descritores/diretórios de arquivos abertos.

CLONE_SIGHAND

- O processo filho e pai compartilham a mesma tabela de manipuladores de sinal.

CLONE_VM

- O processo filho e pai compartilham o mesmo espaço de endereçamento de memória, permitindo uma comunicação muito direta entre cada thread, visto que ambos podem acessar e modificar as mesmas variáveis e estruturas de dados da memória.

Resumo do código

O processo pai aloca uma pilha de memória para o novo processo e o cria usando clone(). O processo filho executará a função threadFunction, enquanto o pai continua a execução logo após a chamada de clone().

O processo pai modifica a variável var_compartilhada para 1, sinalizando ao processo filho. O filho está em um loop de espera ativa (usleep), esperando até que var_compartilhada mude de 0 para 1. Ao detectar essa mudança, imprime uma mensagem de recebimento e altera var_compartilhada para 2.

O processo pai, que estava em um loop de espera ativa (usleep), esperando que seu filho mude seu valor para 2, recebe a confirmação do filho, e dessa forma, o processo pai espera pelo término do processo filho usando waitpid(), e libera os recursos alocados do processo filho, termina assim sua execução.

OBS:

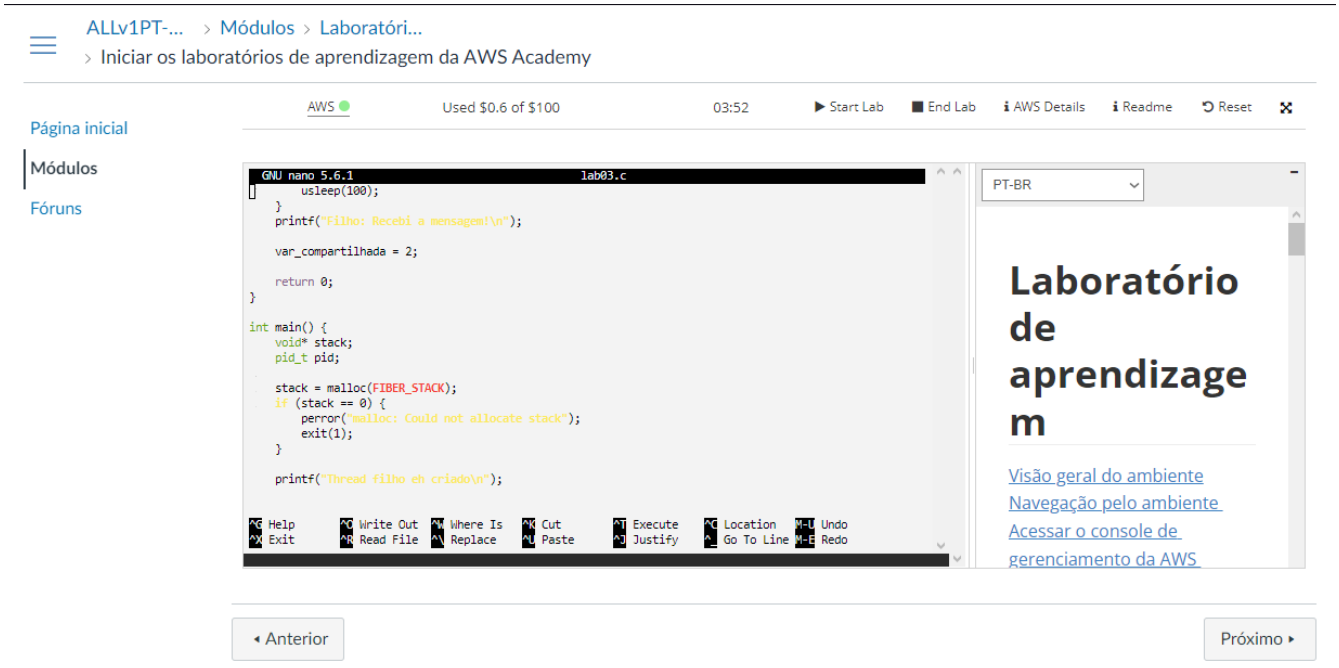
1) Variável Volátil (var_compartilhada): Como é criado um processo que compartilha o mesmo espaço de memória, é permitido a comunicação direta através da modificação de variáveis globais, ou seja, consiste em “dizer” ao compilador não assumir nada sobre o estado dessa variável além do que for explicitamente manipulado pelo programa.

pid = clone(&threadFunction, (char*)stack + FIBER_STACK, SIGCHLD | CLONE_FS | CLONE_FILES | CLONE_SIGHAND | CLONE_VM, 0);

Nesta linha do código, é onde cria-se um novo processo, ou seja, as threads, em que os parâmetros passados consistem em:

- &threadFunction → a função filho que vai ser executada pelo processo, passada anteriormente;
- (char*)stack + FIBER_STACK → O ponteiro aponta para o topo da pilha da memória alocada para o novo processo;
- SIGCHLD | CLONE_FS | CLONE_FILES | CLONE_SIGHAND | CLONE_VM → São passados os flags que especificam o comportamento do clone();
- 0 → É utilizado para passar dados.

Execução do código no editor de texto (nano) dentro do servidor da AWS:



Execução do código com gcc dentro do servidor da AWS:

Referências:

https://eli.thegreenplace.net/2018/launching-linux-threads-and-processes-with-clone/