

EP1 - MAC0422 - Sistemas Operacionais

Sabrina Araujo da Silva - nºUSP 12566182

Samantha Miyahira - nºUSP 11797261

1) Criando uma Shell usando chamadas de sistema

- **Link para acessar a imagem .ova:** https://drive.google.com/drive/folders/1FdiG-YB8dmSiDQoj_9oyyOhauPE7CAdA?usp=sharing

A shell simplificada foi desenvolvida na linguagem C e implementa um programa que recebe comandos e caminhos de arquivo para realizar suas operações. O programa possui quatro funções: *nem_eu_nem_de_ninguem*, *soh_eumesmo*, *rodaeolhe* e *sohroda*.

O programa tem como base um loop infinito que lê comandos e argumentos, compara os comandos com as funções citadas anteriormente e executa a função correspondente. Se o comando digitado pelo usuário não for reconhecido, o programa exibirá uma mensagem informando que o comando é desconhecido e o loop será encerrado, como por exemplo "abc abc".

nem_eu_nem_de_ninguem() e *soh_eumesmo()*

As funções `nem_eu_nem_de_ninguem` e `soh_eumesmo` utilizam a chamada de sistema `chmod` para alterar as permissões dos arquivos especificados. A função `nem_eu_nem_de_ninguem` altera suas permissões para 000, ou seja, nenhum usuário terá permissão de leitura, gravação ou execução. A função `soh_eumesmo` altera suas permissões para 0700, ou seja, apenas o usuário atual terá permissão de leitura, gravação e execução.

rodaeolhe()

A função cria um novo processo usando a função `fork()`. Se ocorrer um erro na criação do processo filho (`pid < 0`), a função `perror()` é chamada para imprimir uma mensagem de erro. Se o processo em execução é o processo filho (`pid == 0`), o programa passado como parâmetro é executado usando a função `execve()`. Se o processo em execução for o processo pai (`pid > 0`), o código aguarda que o processo filho seja encerrado usando a função `wait()` e, em seguida, exibe uma mensagem informando o código de saída do programa.

sohroda()

A função cria um novo processo usando a função `fork()`. Se ocorrer um erro na criação do processo filho (`pid < 0`), a função `perror()` é chamada para imprimir uma mensagem de erro. Se o processo em execução é o processo filho (`pid == 0`), antes da execução do programa, a função `close(STDIN_FILENO)` fecha o descritor de arquivo de entrada padrão (`stdin`) do processo filho e, em seguida, o programa passado como parâmetro é executado usando a função `execve()`.

2) Exemplos de execução

Arquivo usado como teste para alterar as permissões:

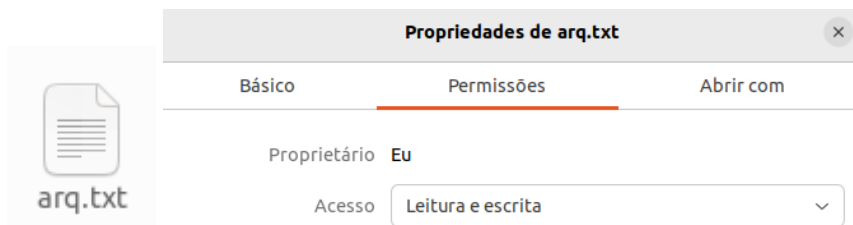


```
./minhaMiniShell  
nem_eu_nem_de_ninguem arq.txt
```



O arquivo "arq.txt" não tem permissão de leitura, escrita ou execução para o usuário dono do arquivo, para o grupo do arquivo e para outros usuários.

```
soh_eumesmo arq.txt
```



Agora, o arquivo "arq.txt" tem permissão total (leitura, escrita e execução) apenas para o usuário dono do arquivo.

Para testar as funções `rodaeolhe()` e `sohroda()` utilizamos um programa chamado **hora** em C que imprime a hora atual do sistema a cada 5 segundos, em um loop que será executado 5 vezes:

```
int main(){
    int contador = 0;
    while(contador < 5){
        time_t t = time(NULL);
        struct tm tm = *localtime(&t);
        printf("Hora atual: %02d:%02d:%02d\n", tm.tm_hour, tm.tm_min, tm.tm_sec);
        sleep(5);
        contador++;
    }

    return 0;
}
```

Usando a minhaMiniShell para executar esse programa e testar a função `rodaeolhe()`, tem-se:

```
rodaeolhe hora
Hora atual: 18:22:10
Hora atual: 18:22:15
Hora atual: 18:22:20
Hora atual: 18:22:25
Hora atual: 18:22:30
Programa hora retornou com codigo 0.
```

Um novo processo filho foi criado para executar o programa **hora**. Após o processo filho terminar a sua execução, o código de retorno 0 foi retornado, ou seja, a execução foi bem-sucedida. Durante a execução do programa em segundo plano, a shell fica bloqueada, aguardando a finalização do programa para poder enviar novos comandos.

Agora, para testar a função `sohroda()`:

```
sohroda hora
Hora atual: 18:25:20
Hora atual: 18:25:25
nem_eu_nem_de_ninguem arq.txt
Hora atual: 18:25:30
Hora atual: 18:25:35
soh_eumesmo arq.txt
Hora atual: 18:25:40
```

Com o comando `sohroda`, é possível executar outros comandos enquanto a shell está executando um programa em background, ou seja, sem bloquear o terminal para o usuário. Para isso, a shell deve criar um novo processo filho e substituir o código do processo filho pelo código do programa

indicado. O processo filho, agora com o código do programa, vai ser executado em background. A shell continua executando em primeiro plano e mostrando a saída na tela, sem bloquear o terminal para o usuário. Enquanto isso, o programa executado em background imprime sua saída na tela, sem interferir na shell. Nesse caso, o programa **hora** continua mostrando a sua saída enquanto outros comandos como "nem_eu_nem_de_ninguem arq.txt" e "soh_eumesmo arq.txt" podem ser executados.

3) Passos para executar o programa no VirtualBox

Inicialmente importar o arquivo .ova no VirtualBox. Após isso, executar a máquina virtual e logar como `root`. Em `/usr/local/src` encontra-se o programa em C ***minhaMiniShell.c*** e em `/usr/local/bin` está o executável ***minhaMiniShell*** deste programa.

Para executar o programa, vá até a pasta `/usr/local/bin` e digite `./minhaMiniShell`. Foram realizados vários testes e alguns deles constam no item 2).

4) Observações

Versão do VirtualBox: VirtualBox 6.1

Versão da ISO: minix 3.1.2a

Sistema Operacional: Ubuntu 20.04.6 LTS