

Universidade Federal de Viçosa - *Campus* Rio Paranaíba

Disciplina: SIN 351 - Sistemas Operacionais

Professor: Rodrigo Moreira

Projeto: Interpretador Shell

Integrantes do projeto:

Amanda Oliveira Nascimento - 5149

Thalita Mendonça Antico - 5141

Vinicius Hideyuki Ikehara -5191

Neste trabalho, criamos um shell, interpretador de comandos, que é responsável por interpretar as instruções enviadas pelo usuário e programas ao sistema operacional. Executa comandos lidos do teclado ou de um arquivo executável, sendo a principal ligação entre o usuário, os programas e o sistema operacional.

Temos as chamadas de sistemas **Fork()**, que cria uma cópia quase idêntica do processo atual.

O **execvp()** consiste em uma função que provê um vetor de ponteiros para strings não nulas, reproduzindo a lista de argumentos para o programa.

Wait(), consistem em chamadas que possibilitam que o processo pai aguarde por seus filhos e adquira seu código de terminação, nesse caso deixar pendente a execução do processo até o término (fim) de seu filho. Caso seu filho já esteja finalizado, no instante em que é feito a chamada, a função retorna em seguida.

Iniciando o Shell:

No nosso .zip temos os arquivos "makefile" e "shell.c".

Para compilar o código você deve abrir a pasta onde está localizado os arquivos "makefile" e "shell.c".

Após abrir a pasta será necessário abrir o terminal dentro deste diretório, do contrário o código não será encontrado pelo terminal.

Com o terminal aberto no diretório correto, agora para compilar o código é só digitar no terminal a palavra "make" e apertando o enter.

Se tudo tiver dado certo o seu terminal não irá acusar nenhum erro e agora é possível executar o código.

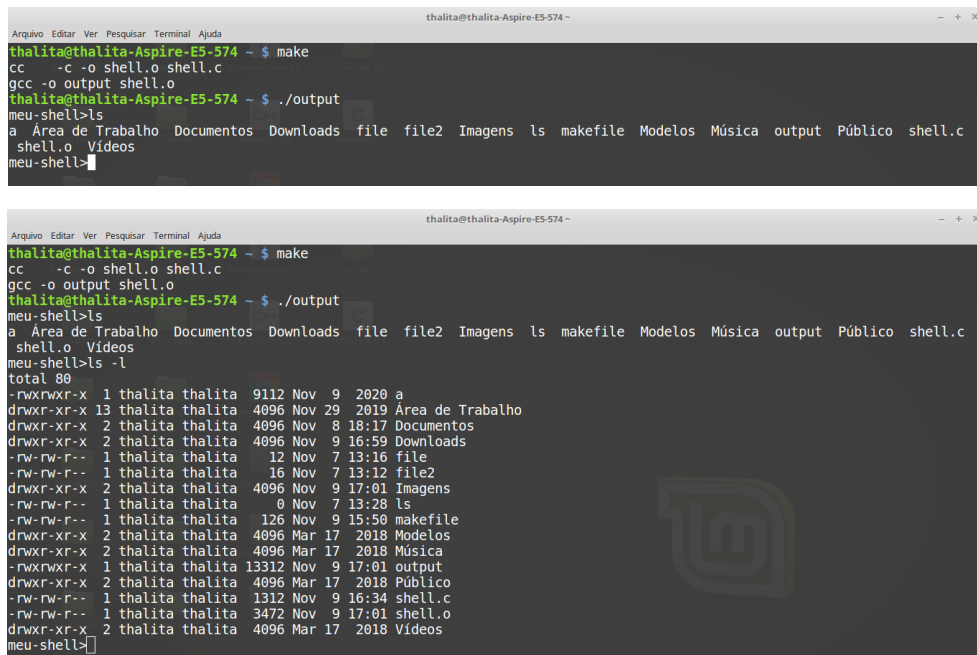
Para executar o código é necessário digitar no terminal `./output` e apertar o enter.

Shell em execução:

Se após os processos de inicialização no seu terminal ao invés do seu user Linux estiver aparecendo `meu-shell>` então é porque agora você já está utilizando o nosso Interpretador Shell e pode começar a explorar e digitar comandos para o terminal Linux.

Alguns comandos testados que obtiveram êxito:

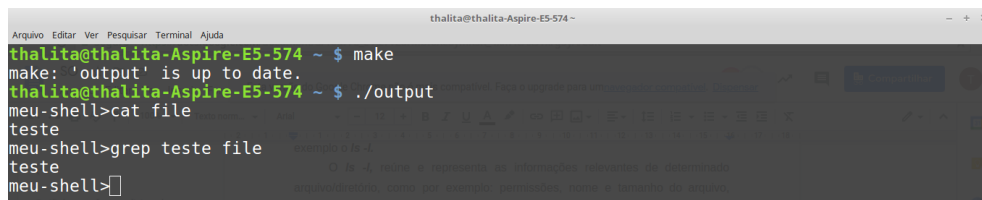
`"ls"`:



```
thalista@thalita-Aspire-E5-574 ~$ make
cc -c -o shell.o shell.c
gcc -o output shell.o
thalista@thalita-Aspire-E5-574 ~$ ./output
meu-shell>ls
a Área de Trabalho Documentos Downloads file file2 Imagens ls makefile Modelos Música output Público shell.c
shell.o Vídeos
meu-shell>

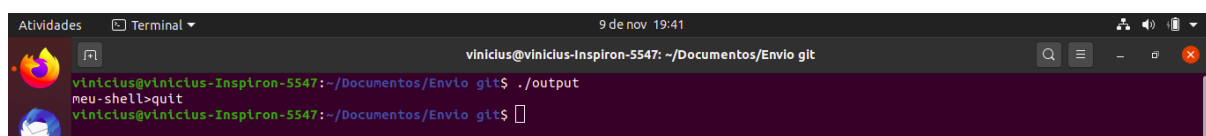
thalista@thalita-Aspire-E5-574 ~$ make
cc -c -o shell.o shell.c
gcc -o output shell.o
thalista@thalita-Aspire-E5-574 ~$ ./output
meu-shell>ls
a Área de Trabalho Documentos Downloads file file2 Imagens ls makefile Modelos Música output Público shell.c
shell.o Vídeos
meu-shell>ls -l
total 80
-rwxrwxr-x 1 thalita thalita 9112 Nov  9 2020 a
drwxr-xr-x 13 thalita thalita 4096 Nov 29 2019 Área de Trabalho
drwxr-xr-x  2 thalita thalita 4096 Nov  8 18:17 Documentos
drwxr-xr-x  2 thalita thalita 4096 Nov  9 16:59 Downloads
-rw-rw-r-- 1 thalita thalita  12 Nov  7 13:16 file
-rw-rw-r-- 1 thalita thalita  16 Nov  7 13:12 file2
drwxr-xr-x  2 thalita thalita 4096 Nov  9 17:01 Imagens
-rw-rw-r-- 1 thalita thalita   0 Nov  7 13:28 ls
-rw-rw-r-- 1 thalita thalita 126 Nov  9 15:50 makefile
drwxr-xr-x  2 thalita thalita 4096 Mar 17 2018 Modelos
drwxr-xr-x  2 thalita thalita 4096 Mar 17 2018 Música
-rwxrwxr-x 1 thalita thalita 13312 Nov  9 17:01 output
drwxr-xr-x  2 thalita thalita 4096 Mar 17 2018 Público
-rw-rw-r-- 1 thalita thalita 1312 Nov  9 16:34 shell.c
-rw-rw-r-- 1 thalita thalita 3472 Nov  9 17:01 shell.o
drwxr-xr-x  2 thalita thalita 4096 Mar 17 2018 Vídeos
meu-shell>
```

`"grep" e "cat"`:



```
thalista@thalita-Aspire-E5-574 ~$ make
make: 'output' is up to date.
thalista@thalita-Aspire-E5-574 ~$ ./output
meu-shell>cat file
teste
meu-shell>grep teste file
teste
meu-shell>
```

`"quit"`:

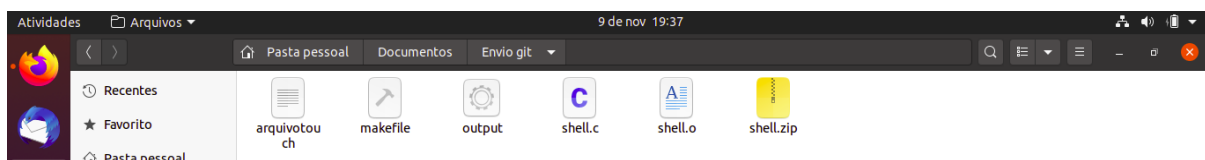
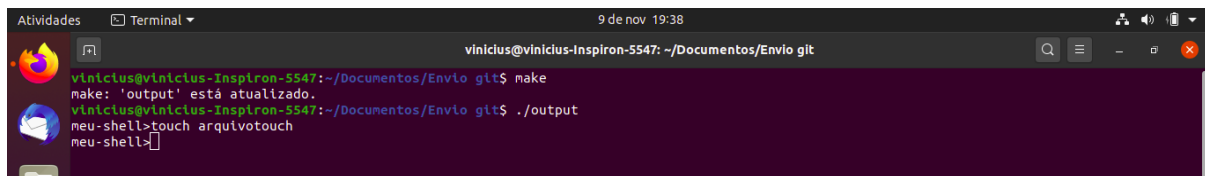
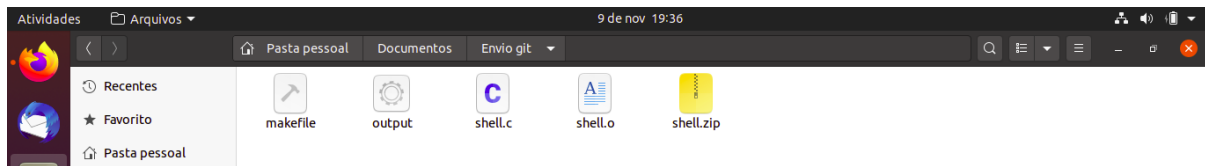


```
vinicius@vinicius-Inspiron-5547: ~/Documentos/Envio git$ ./output
meu-shell>quit
vinicius@vinicius-Inspiron-5547:~/Documentos/Envio git$
```

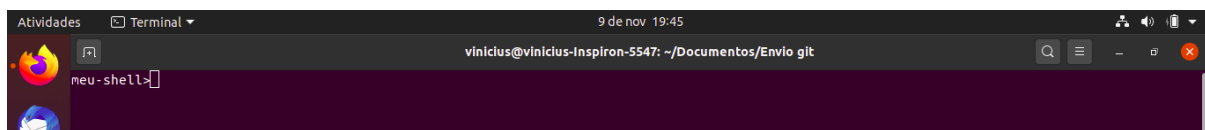
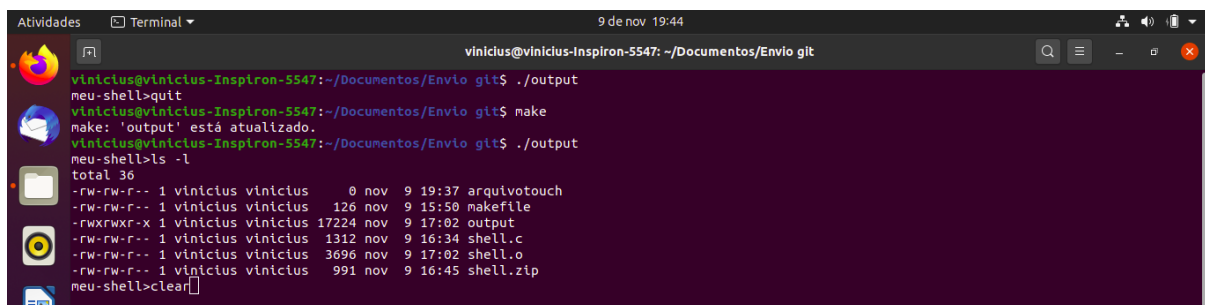
`"poweroff"`:

Esse comando não tem como printar porque ele desliga o computador.

"touch";



“clear”;

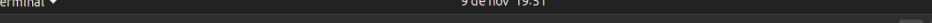


PS: Este foram os comandos conhecidos e testados por nós, testamos apenas estes pois são os comandos que conhecemos.

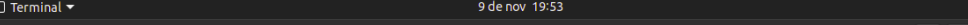
Na descrição do trabalho disponibilizada pelo professor Rodrigo, o nosso Shell deveria ser capaz de interpretar comandos concatenados por ",", exemplo: ls -l, cat file1.

No entanto não obtivemos êxito com relação aos comandos concatenados com ",", o máximo que conseguimos fazer foi comandos que começam com ",", e que não temos nenhum espaço em branco do tipo " ".

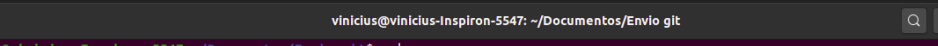
Portanto considerando a aceitação de comandos com ",", os comandos testados que obtiveram êxito foram os que começam com a “,”:



```
meu-shell>quit
vinicius@vinicius-Inspiron-5547:~/Documentos/Envio git$ make
make: 'output' está atualizado.
vinicius@vinicius-Inspiron-5547:~/Documentos/Envio git$ ./output
meu-shell>,ls
arquivotouch makefile output shell.c shell.o shell.zip
meu-shell>
```



The screenshot shows a terminal window with a dark background. The title bar at the top reads "Atividades" and "Terminal". The current directory is "~/Documentos/Envlo git". The prompt is "vinicius@vinicius-Inspiron-5547: ~/Documentos/Envlo git". The user has entered the command `./output`, which has executed successfully, displaying the output `meu-shell>,quit`. The prompt is now `vinicius@vinicius-Inspiron-5547: ~/Documentos/Envlo git$`.



The screenshot shows a terminal window with a dark background. The title bar at the top reads 'Atividades' and 'Terminal'. The prompt is 'vinicius@vinicius-Inspiron-5547: ~/Documentos/Envlo git'. The user has entered the command 'make', and the output is 'make: 'output' está atualizado.'. The user then enters './output', and the prompt changes to 'meu-shell>'. The terminal window has standard Ubuntu window controls (minimize, maximize, close) and a search icon in the top right corner.

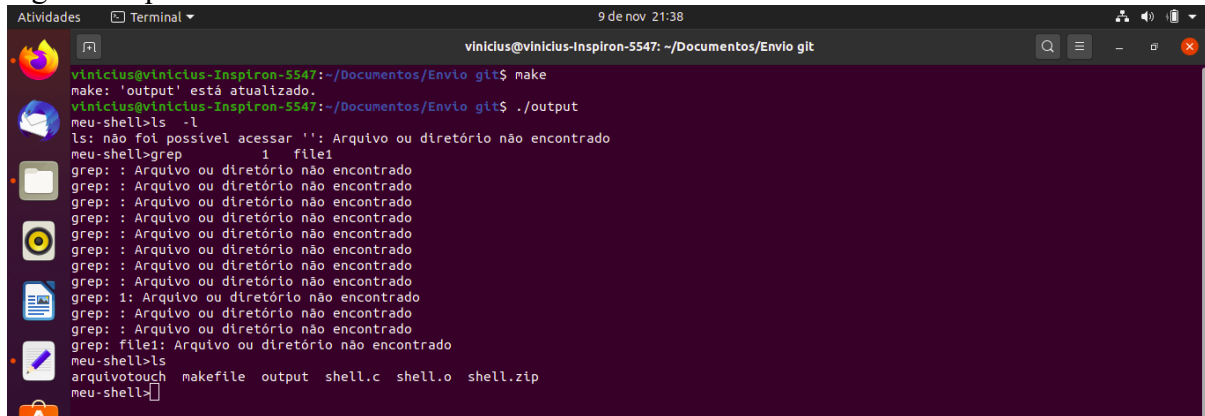
```
vinicius@vinicius-Inspiron-5547: ~/Documentos/Envlo git
make: 'output' está atualizado.
meu-shell>
```

[illegible]

Input com 514 caracteres:

Espaços extra entre comandos deverão ser desconsiderados: infelizmente essa situação conseguiu ser tratada e caso um comando tenha espaços aparecem as mensagens normais de erro porém o terminal não é encerrado ou crashado e você pode digitar um outro comando

logo na sequência:



```
vinicius@vinicius-Inspiron-5547:~/Documentos/Envio git$ make
make: 'output' está atualizado.
vinicius@vinicius-Inspiron-5547:~/Documentos/Envio git$ ./output
meu-shell$ ls -l
ls: não foi possível acessar '': Arquivo ou diretório não encontrado
meu-shell$ grep 1 file1
grep: : Arquivo ou diretório não encontrado
grep: : Arquivo ou diretório não encontrado
grep: : Arquivo ou diretório não encontrado
grep: : Arquivo ou diretório não encontrado
grep: : Arquivo ou diretório não encontrado
grep: : Arquivo ou diretório não encontrado
grep: : Arquivo ou diretório não encontrado
grep: : Arquivo ou diretório não encontrado
grep: 1: Arquivo ou diretório não encontrado
grep: : Arquivo ou diretório não encontrado
grep: : Arquivo ou diretório não encontrado
grep: file1: Arquivo ou diretório não encontrado
meu-shell$ ls
arquivotouch makefile output shell.c shell.o shell.zip
meu-shell$
```

Não foram encontradas situações em que o interpretador teve que ser encerrado forçadamente apertando o “x” no canto superior direito em sua última versão, porém nas primeiras versões houveram momentos que digitando uma certa sequência de códigos, o shell travava num loop em que não importava o que fosse digitado que nenhum comando seria computado, nem mesmo o quit.