MC202 – Aula de Laboratório

Ambiente de Desenvolvimento Linux

Ambiente Virtual

Durante o desenvolvimento desta disciplina iremos utilizar um ambiente GNU/Linux.

O VirtualBox é um software de virtualização que permite instalar e executar diferentes sistemas operacionais em um único computador sem complicações. Com ele, o usuário pode executar o Linux dentro do Windows ou Mac.

- workspace32bits

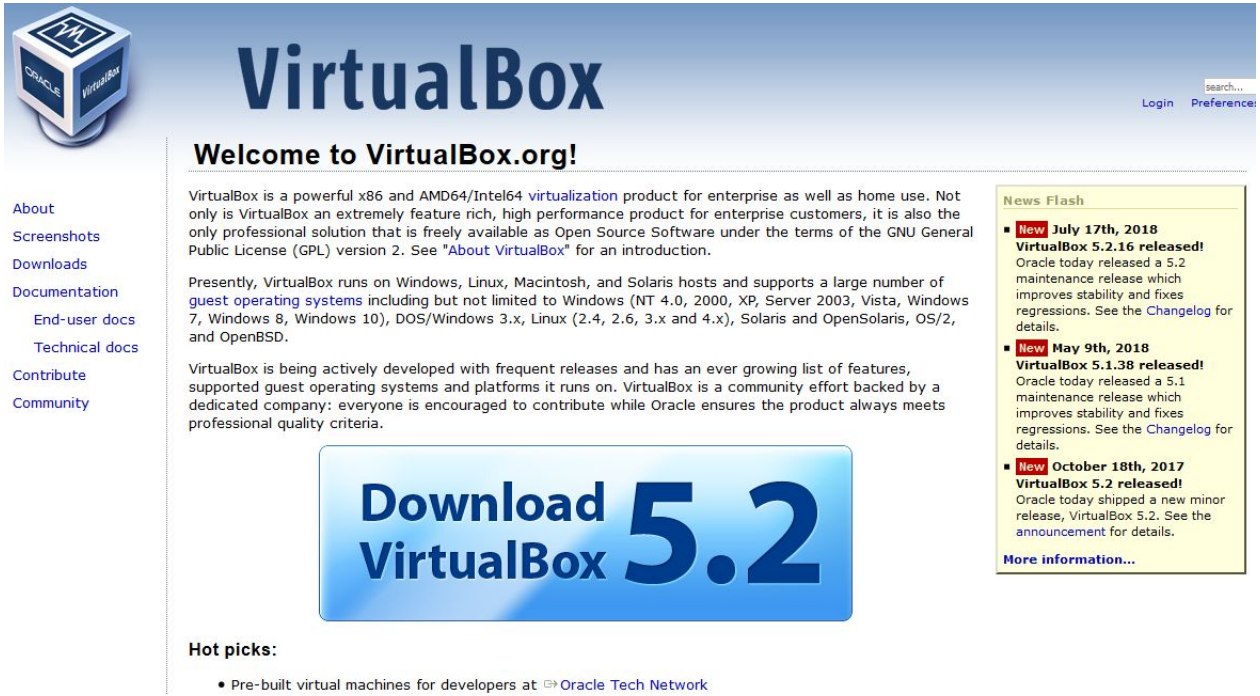
Instalação e Configuração do Ambiente Virtual

Na página da disciplina:
(http://www.ic.unicamp.br/~rafael/cursos/2s2018/mc202)

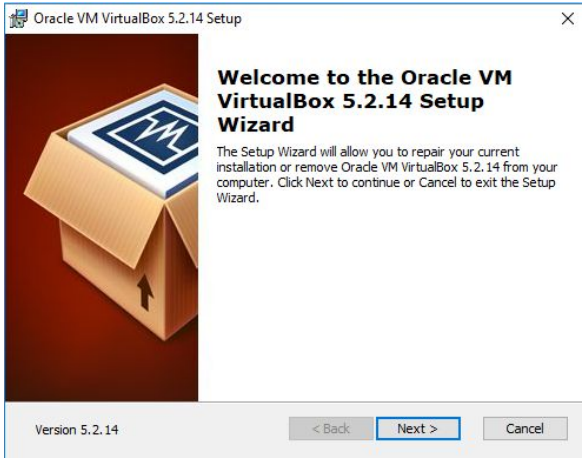
Na seção “Links Úteis/Interessantes”, clique em “Máquina Virtual Linux” para baixar o arquivo “workspace-32bits.zip” e extraia ele no diretório desejado. Vamos usar esse arquivo mais tarde.

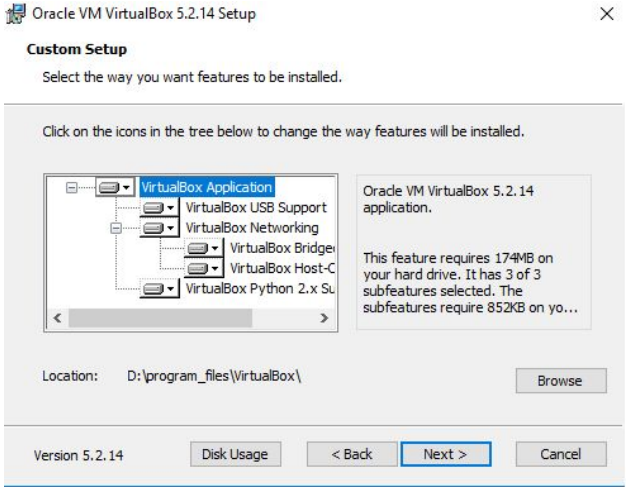
Instalação e Configuração do Ambiente Virtual

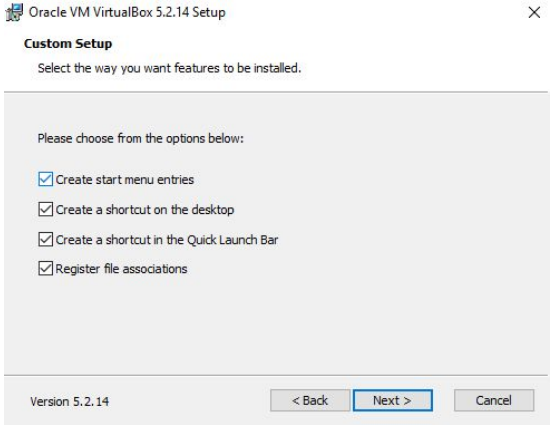
Site: https://www.virtualbox.org/

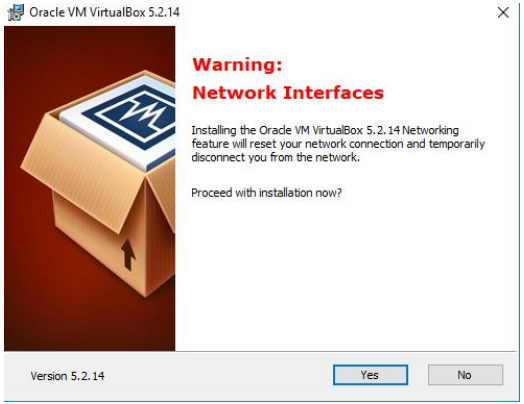


Next->Next->Next->Yes



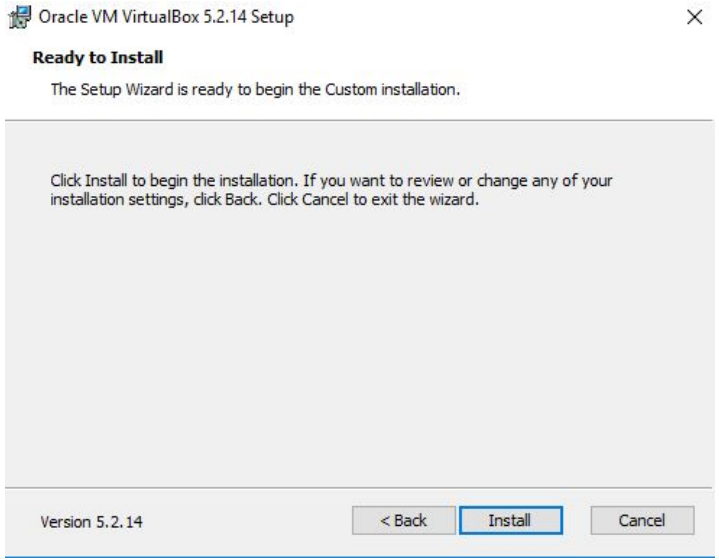


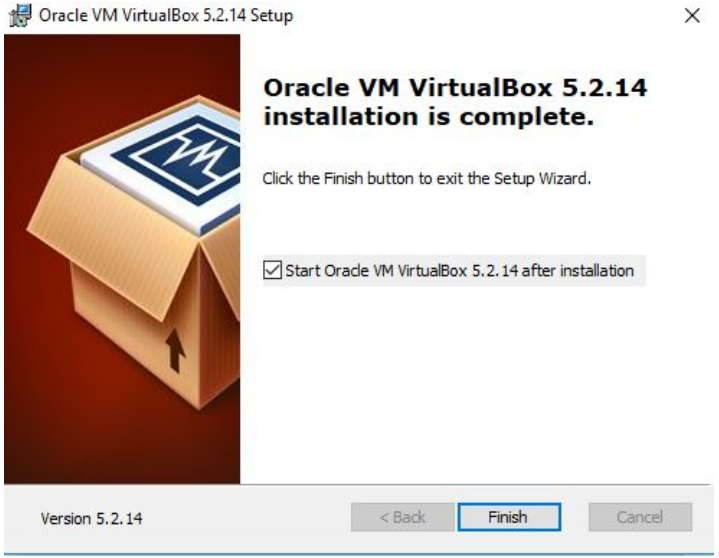




Instalação e Configuração do Ambiente Virtual

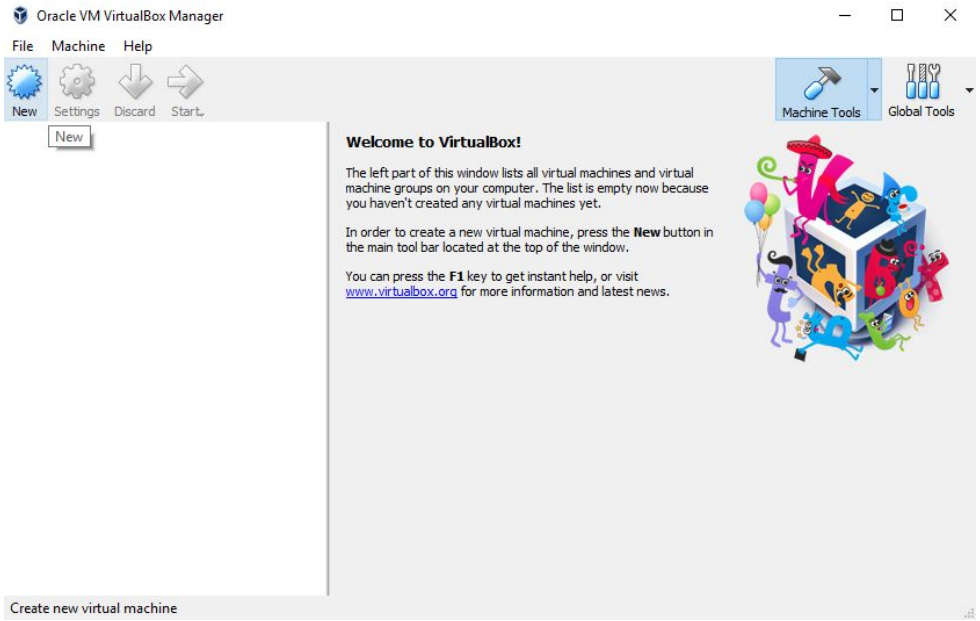
Após apertar o botão para instalar, o programa vai pedir por permissões de administrador. Aceite. Após esse passo o VirtualBox estará instalado.





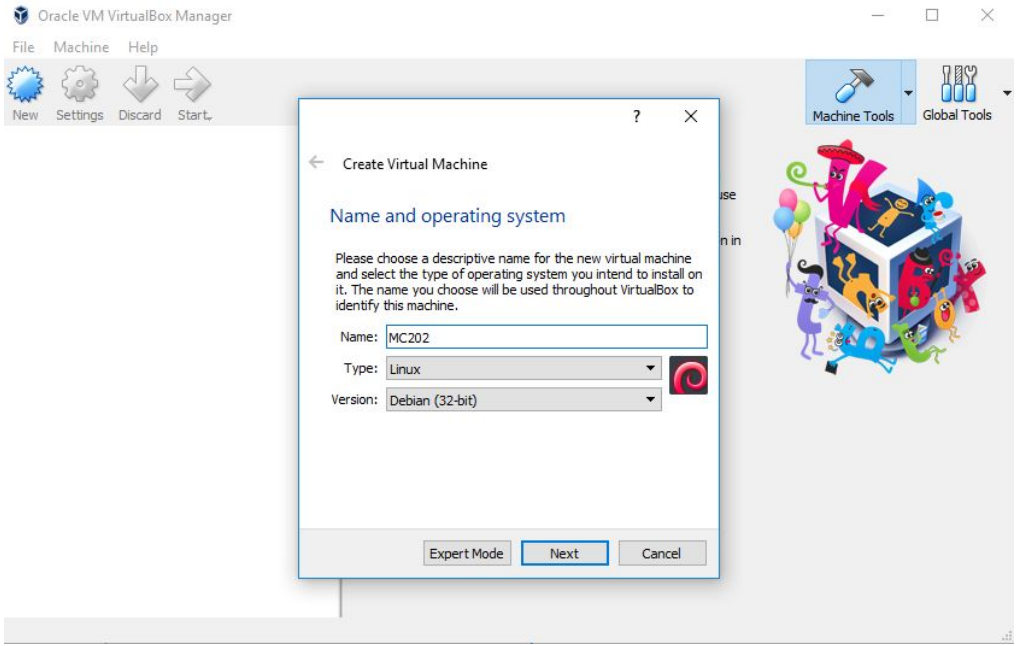
Instalação e Configuração do Ambiente Virtual

Com o programa aberto, clique em “New” para começar a configurar o ambiente virtual.



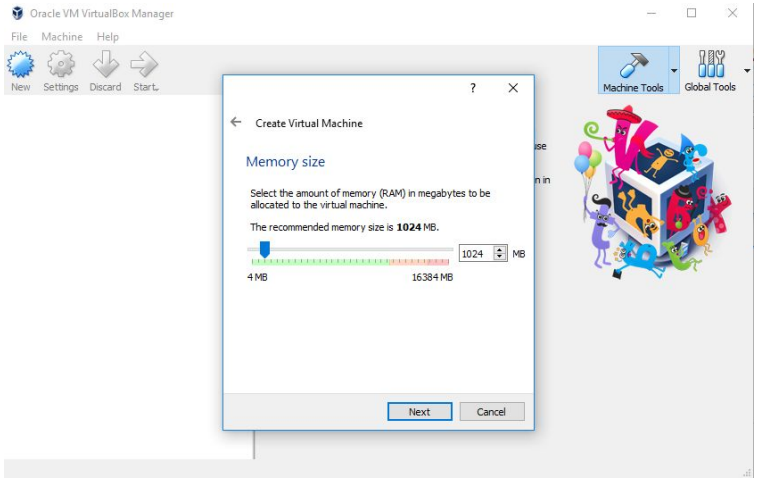
Instalação e Configuração do Ambiente Virtual

As opções na tela seguinte são para identificação do ambiente, preencha da seguinte forma:



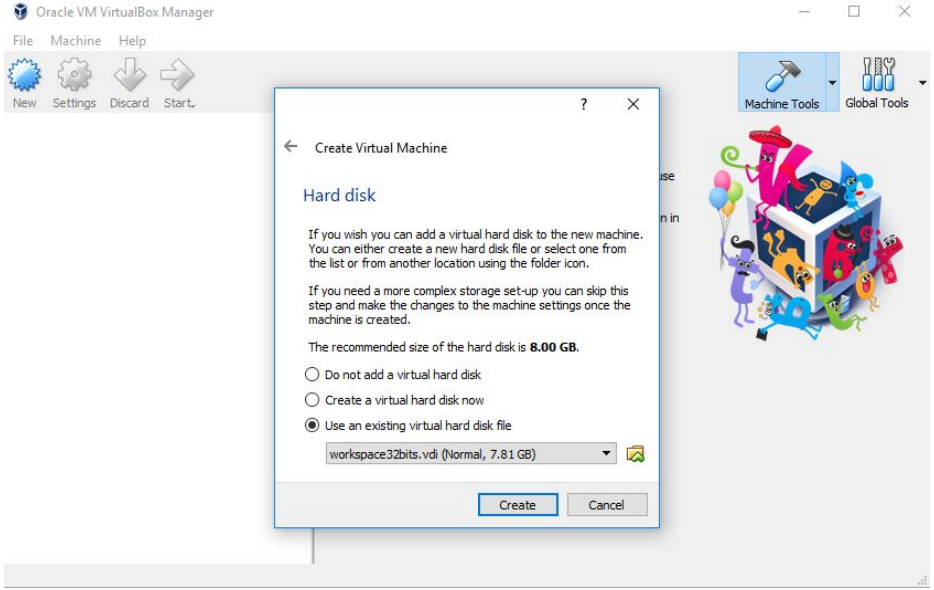
Instalação e Configuração do Ambiente Virtual

Aqui escolhemos quanto de memória RAM o nosso ambiente vai consumir. Quanto mais, maior a performance da nossa máquina. Se o seu computador tiver 4 Gigabytes ou mais de RAM recomendamos deixar um 2048, se não coloque metade da RAM da sua máquina. (Isso pode ser alterado mais tarde)



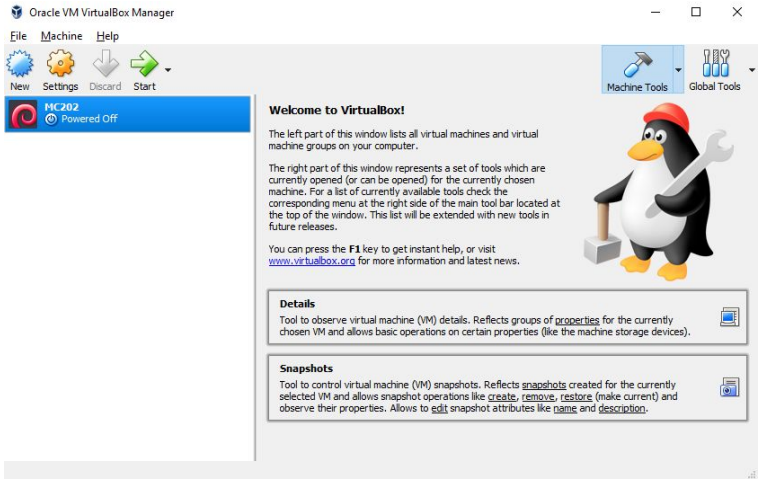
Instalação e Configuração do Ambiente Virtual

Aqui escolhemos a última opção, e no campo em baixo escolhemos o arquivo workspace32bits.vdi que foi baixado da página do professor. Depois clique em “Create”.



Instalação e Configuração do Ambiente Virtual

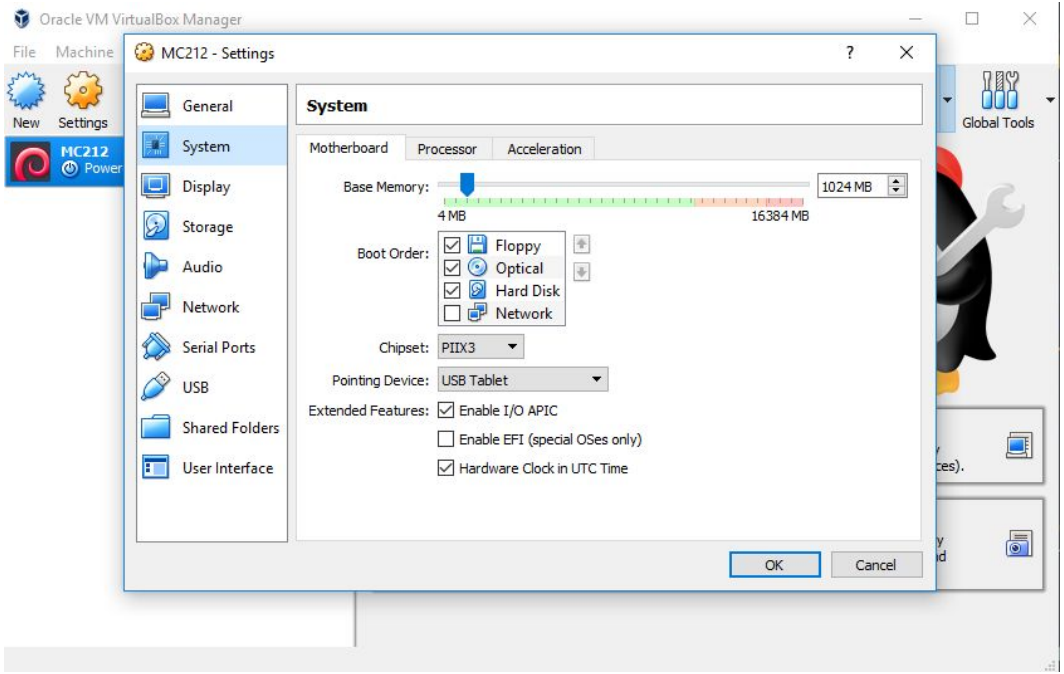
O ambiente virtual deveria estar pronto, clicando em “Start” ele deveria começar, mas ainda é possível que ele falhe, se isso acontecer, existem algumas configurações que ainda podemos arrumar.



Instalação e Configuração do Ambiente Virtual

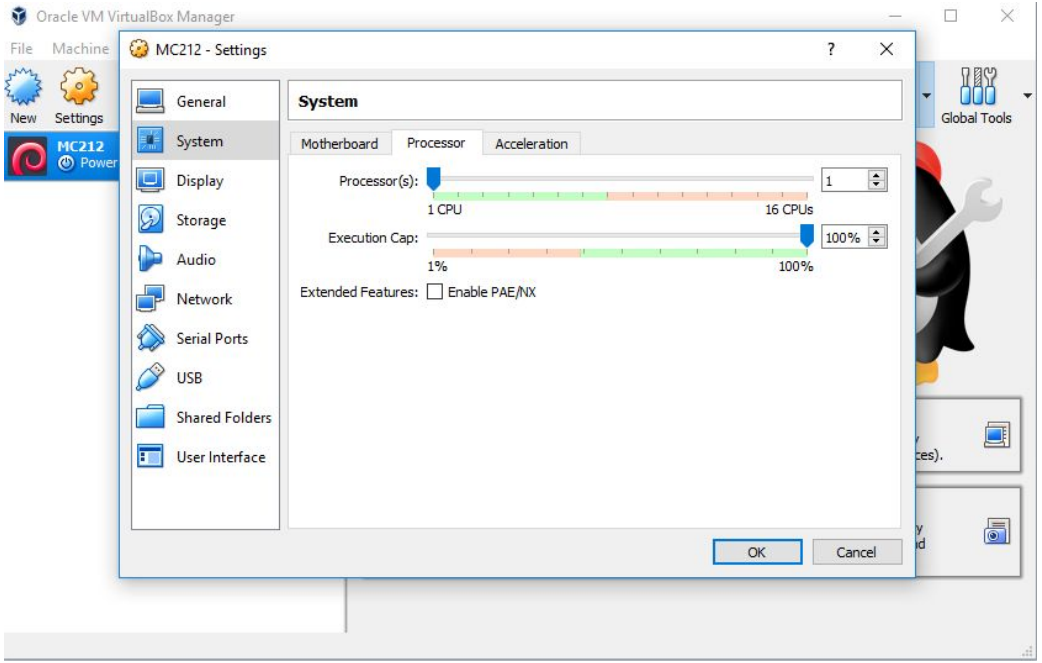
Em “Settings” escolha a aba “System” e habilite a opção “Enable I/O APIC”.

Isso deveria resolver os problemas restantes.



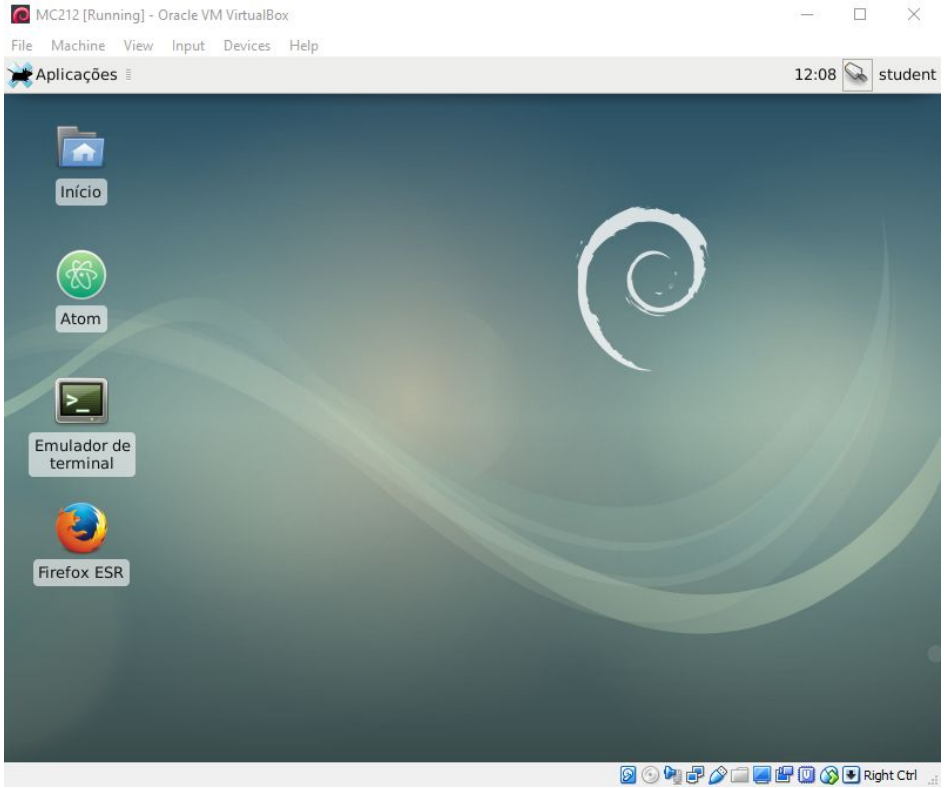
Instalação e Configuração do Ambiente Virtual

Ainda em “System”, na aba “Processor” você pode aumentar a quantidade the processadores simultâneos que serão usados pelo ambiente virtual. Isso aumenta o desempenho dele.



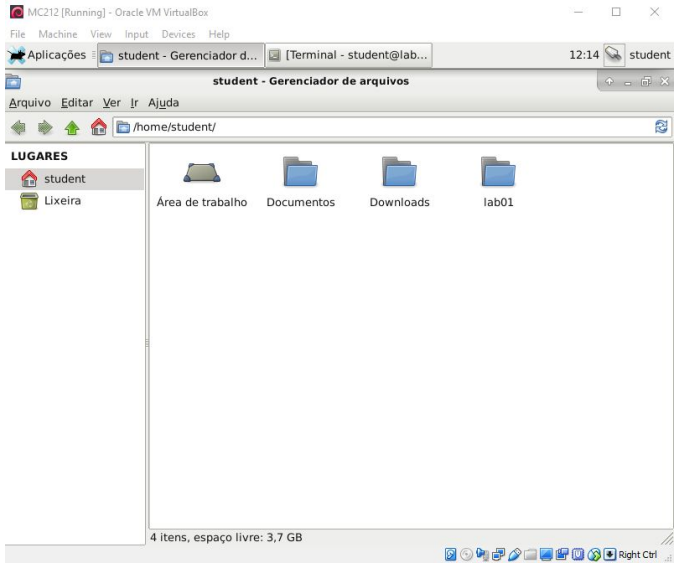
Instalação e Configuração do Ambiente Virtual

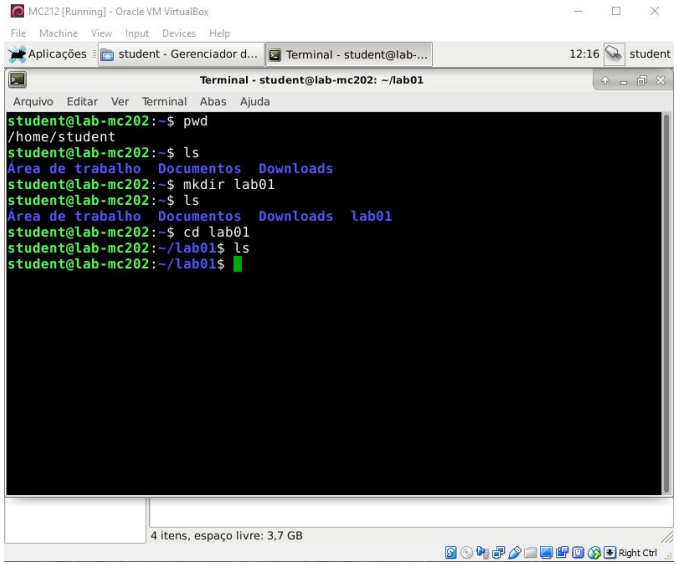
No ambiente você tem acesso ao “Firefox” para conseguir baixar os arquivos dos laboratórios.
O editor “Atom” para escrever o código.
E um “Emulador de terminal”.



Emulador de Terminal

O terminal pode ser visto como uma interface semelhante a de um “gerenciador de arquivos” (que pode ser aberto pelo atalho “início”).

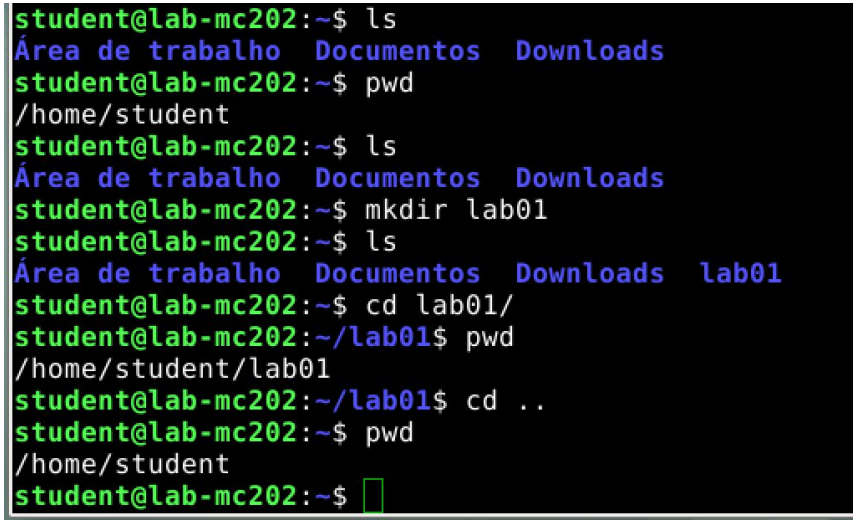




Emulador de Terminal

Alguns comandos:

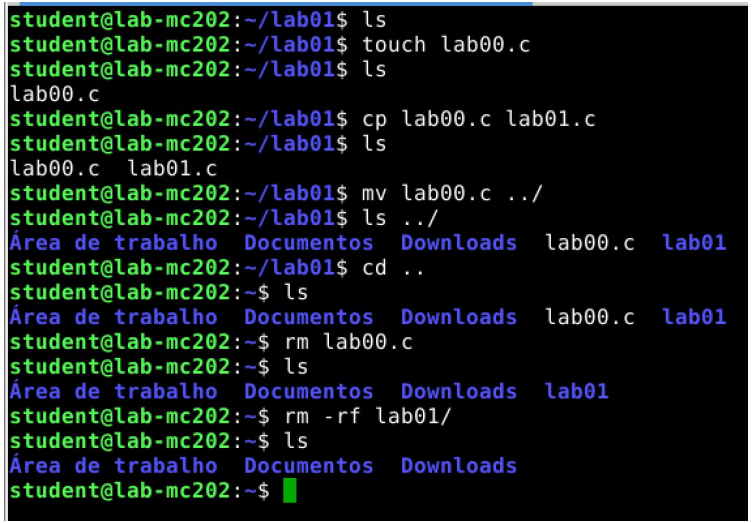
- **pwd** - Mostra o caminho do diretório atual
- **ls** - Lista os arquivos dentro do diretório atual
- **mkdir** <diretorio> - Cria um diretório
- **cd** <diretorio> - Entra no diretório
- **cd ..** - Volta para o diretório anterior



Emulador de Terminal

Alguns comandos:

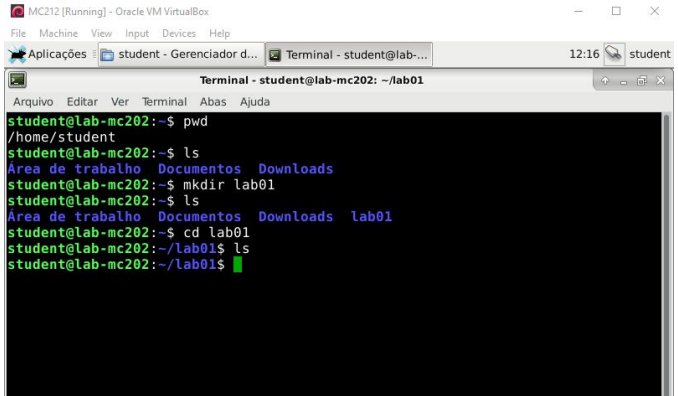
- **touch** <arquivo>- cria um arquivo vazio
- **cp** <arquivo> <copia> - cria uma cópia de um arquivo
- **mv** <arquivo> <dir> - move arquivo para diretório destino
- **rm** <arquivo> - remove permanentemente um arquivo
- **rm -rf** <diretorio> - remove um diretório e todos seus conteúdos



Exemplo: Lab01

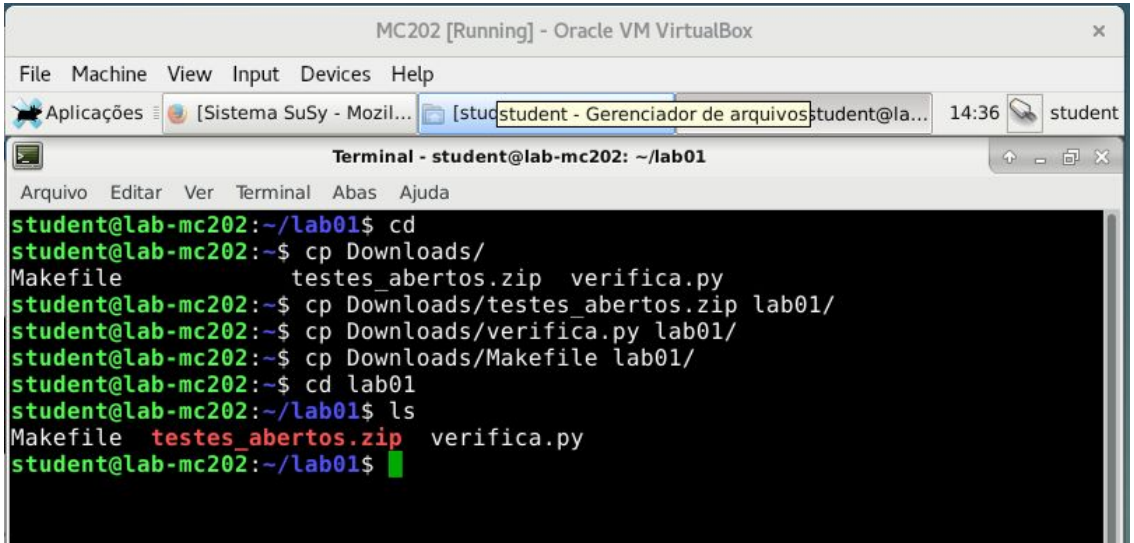
Página do susy:
https://susy.ic.unicamp.br:9999/mc202efgh/SANDBOX-01

Dentro da máquina virtual baixar os arquivos auxiliares: "testes_abertos.zip", "verifica.py" e "Makefile". Crie pelo terminal a pasta "lab01" como indicado na imagem.



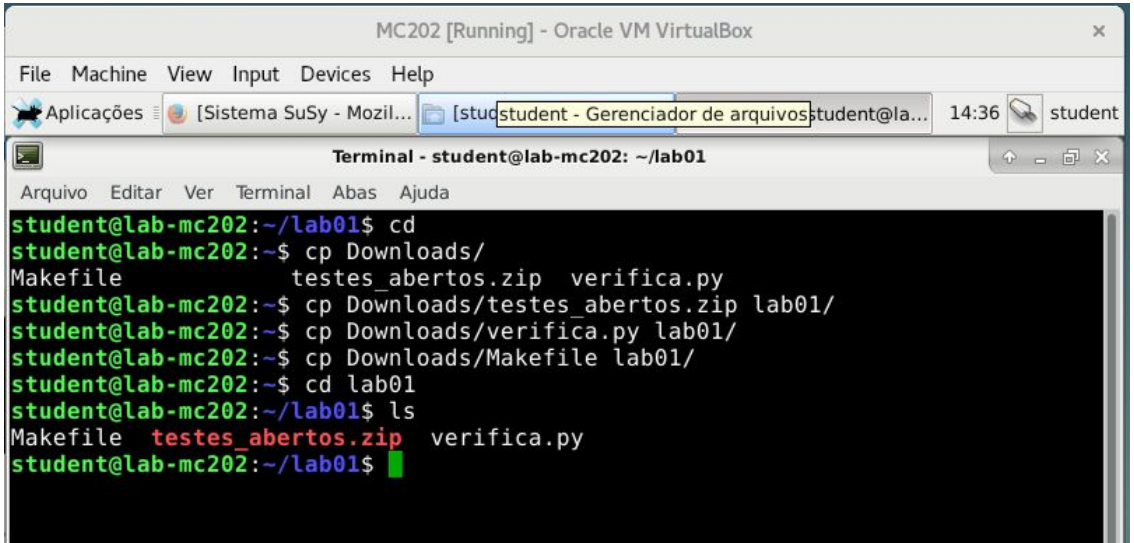
Exemplo: Lab01

Os arquivos baixados vão estar no diretório: /home/student/Downloads.
Vamos mover os arquivos agora para a pasta "lab01" com os seguintes comandos:



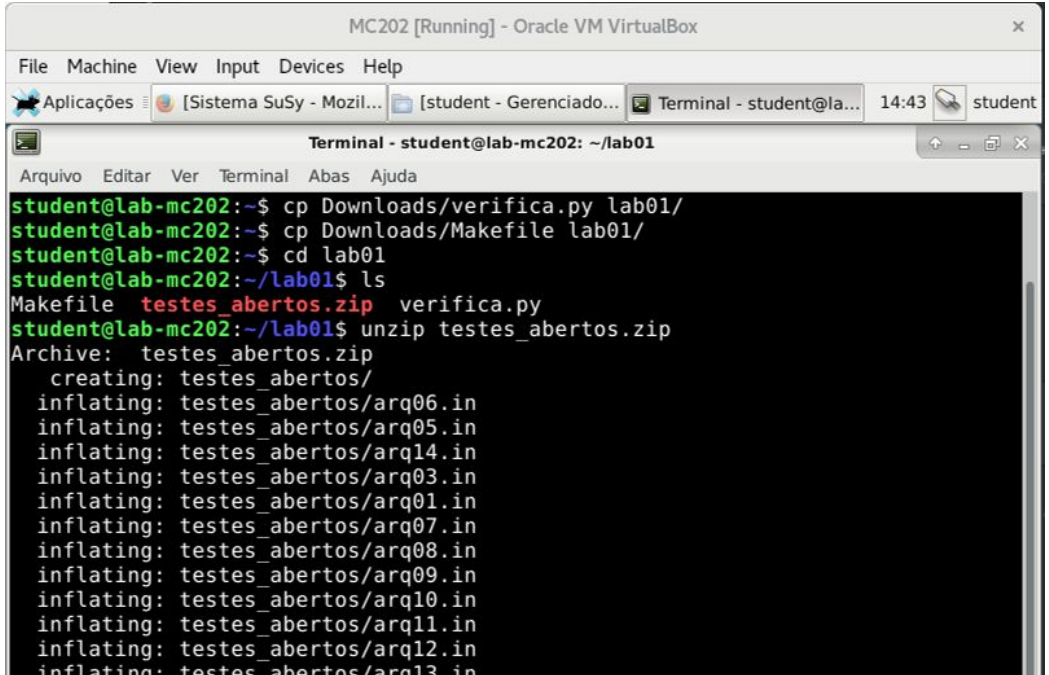
Exemplo: Lab01

- cd - Volta para o diretório /home/students
- cp Downloads/ARQUIVO lab01/ - Copia o ARQUIVO do diretório Download para a pasta "lab01"



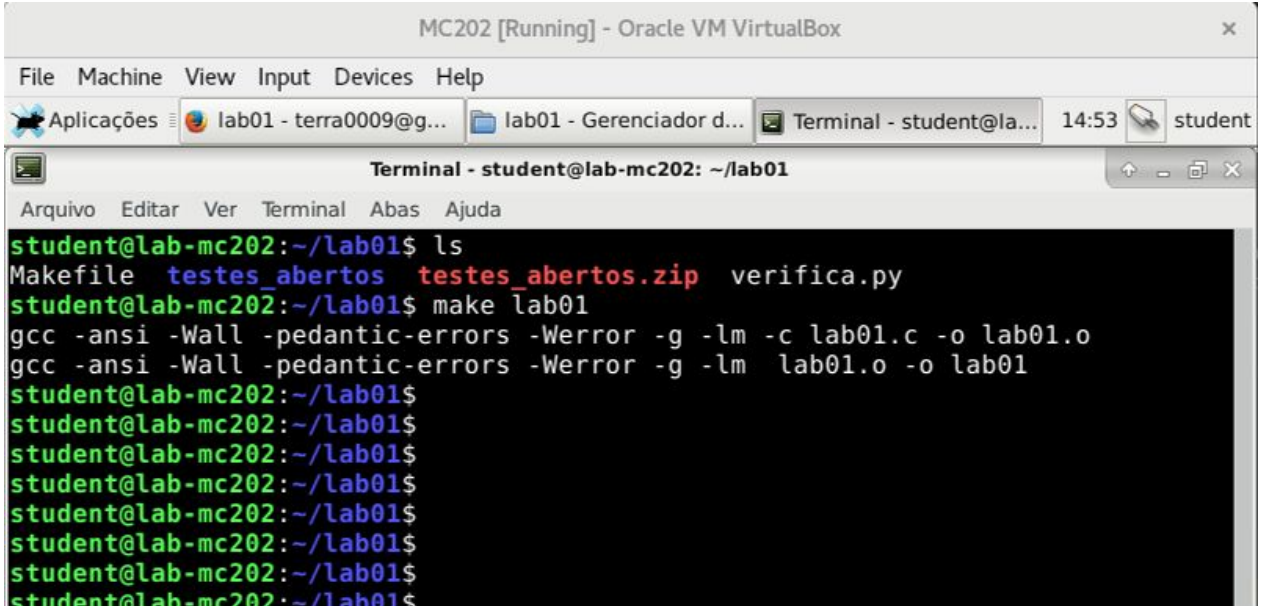
Exemplo: Lab01

- unzip ARQUIVO.zip - Cria uma pasta com o nome ARQUIVO e descomprime o conteúdo de ARQUIVO.zip dentro dessa pasta.



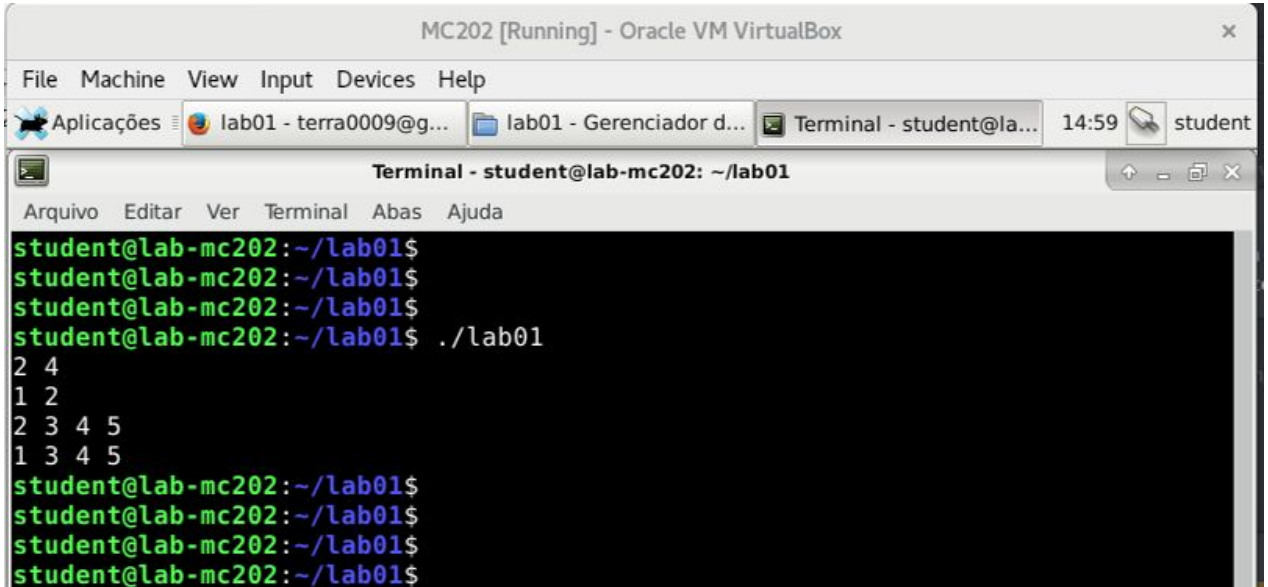
Exemplo: Lab01

make lab01 - O aluno deve criar o arquivo lab01.c que vai conter o código do laboratório. O comando make lab01 compila o código do lab01.c criando o executável "lab01"



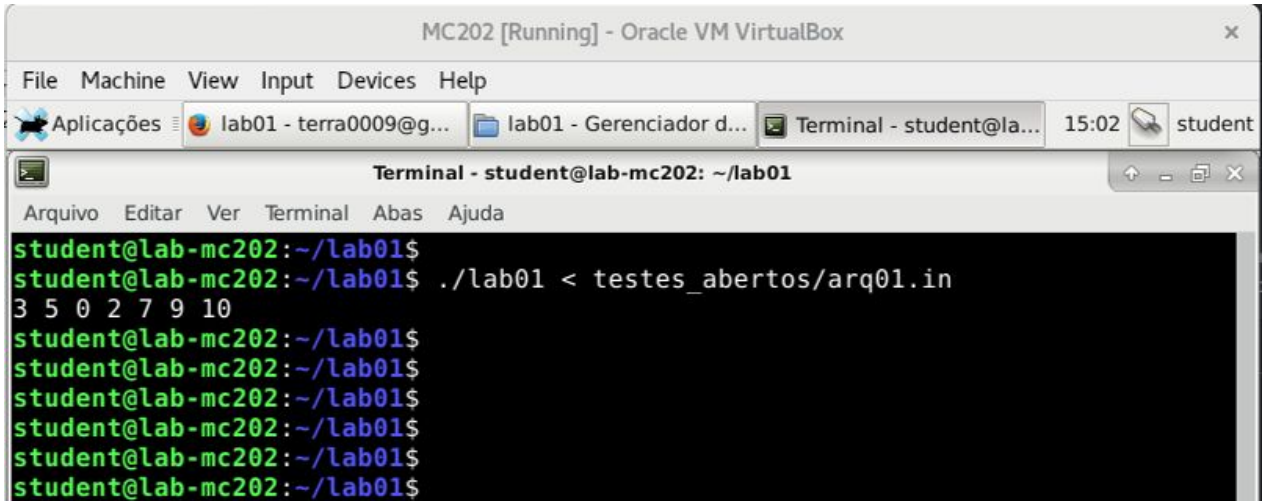
Exemplo: Lab01

./lab01 - Esse comando executa o programa lab01.
Nesse caso, a entrada do programa deve ser especificada pelo próprio usuário como mostrado abaixo.



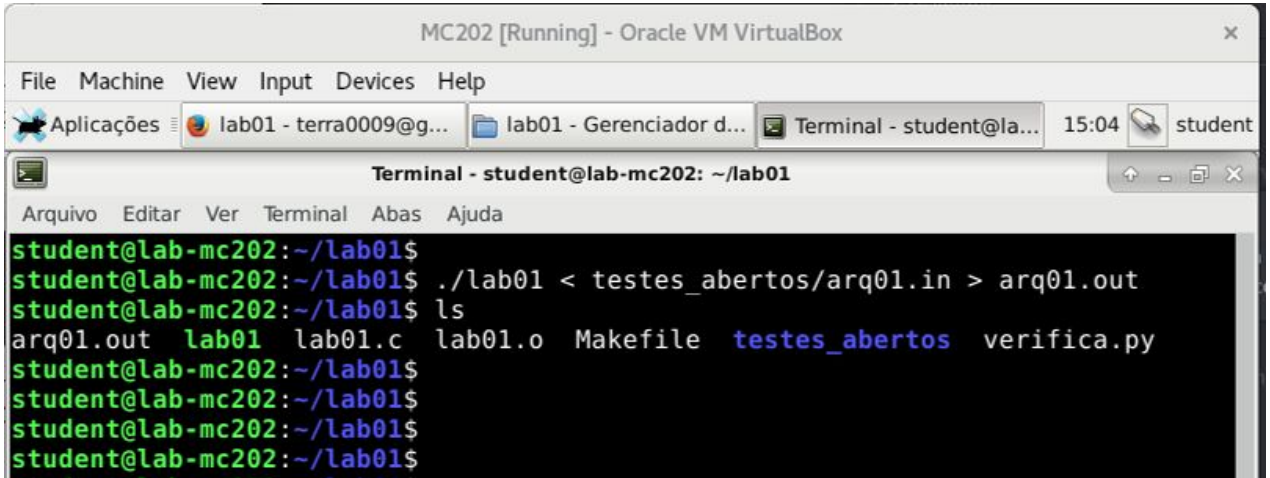
Exemplo: Lab01

./lab01 < testes_abertos/arq01.in - Esse comando executa o programa "lab01", mas ao invés de receber a entrada do teclado, ele recebe como entrada as informações do arquivo arq01.in na pasta "testes_abertos"



Exemplo: Lab01

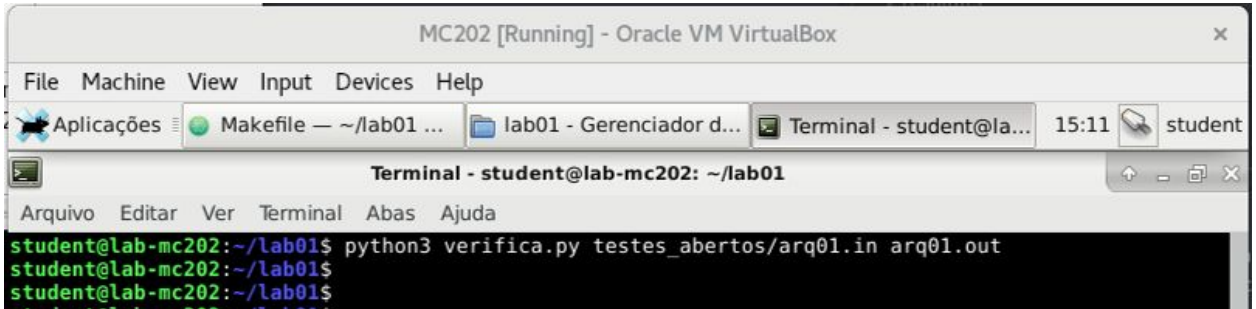
./lab01 < testes_abertos/arq01.in > arq01.out - Esse comando pega a entrada do arquivo "testes_abertos/arq01.in" e ao invés de jogar a saída no terminal, ele joga dentro do arquivo arq01.out. Usando o comando "ls" podemos ver que o arquivo arq01.out foi criado.



Exemplo: Lab01

python3 verifica.py testes_abertos/arq01.in arq01.out

O código em python "verifica.py" é responsavel por verificar se a resposta gerada pelo programa está certo.
Para executá-lo chamamos o interpretador "python3" com os código seguido da entrada e da saída que foi gerada pelo nosso programa. Se não houver saída, quer dizer que o nosso resultado está certo.



Exemplo: Lab01

```
make testar_tudo
```

Esse comando compila o nosso programa, executa ele com cada uma das entradas diferentes e verifica se o resultado está certo. É uma ferramenta que deixa mais fácil o nossos testes.

Makefile

No exemplo do lab01, usamos o comando "make" tanto para compilar quanto para testar. Esse comando age de acordo com as informações dentro do arquivo "Makefile" que é fornecido junto com a atividade.

Nele já tem as instruções para compilar o código da forma desejada, por isso o processo de compilação é tão simples.

GCC

GCC é o verdadeiro programa que compila a nossa tarefa. Ele é chamado quando usamos o comando "make".
O nosso Makefile foi configurado para o comando "make lab01" ser equivalente ao seguinte comando:

```
gcc -ansi -Wall -pedantic-errors -Werror -g -lm lab01.c -o lab01
```

GCC

```
gcc -ansi -Wall -pedantic-errors -Werror -g -lm lab01.c -o lab01
```

-ansi : informa o compilador qual é a variação da linguagem c a ser usada

-Wall : gera avisos quando o compilador suspeita que algo pode estar errado no nosso código

-pedantic-errors : Gera erros adicionais se o código não segue os padrões definidos pela ISO.

GCC

```
gcc -ansi -Wall -pedantic-errors -Werror -g -lm lab01.c -o lab01
```

-Werror : todo aviso gerado pelo compilador é tratado como error

-g : Gera informações de debug (será coberto em outra aula)

-lm : Linka o código com a biblioteca "m" (não é necessário para esse laboratório, mas no futuro será)

GCC

```
gcc -ansi -Wall -pedantic-errors -Werror -g -lm lab01.c -o lab01
```

lab01.c : Nome do arquivo com o código

-o lab01 : Nome do executável que será gerado

Dúvidas?