

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

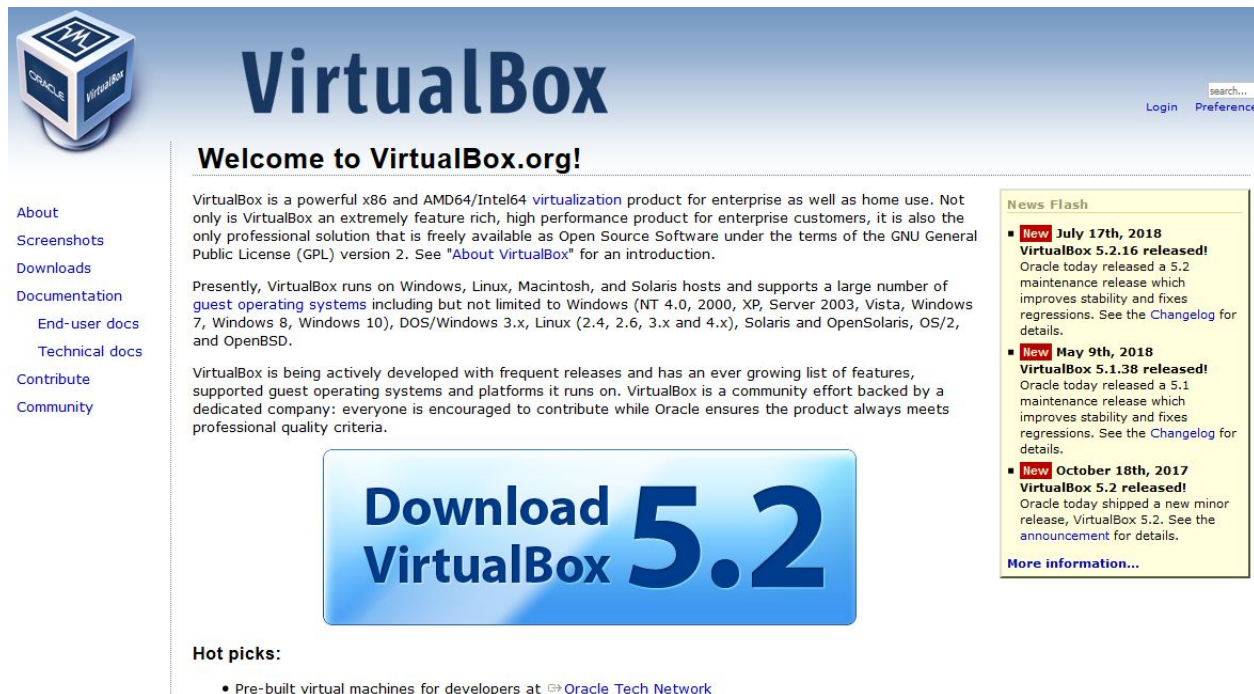
Na página da disciplina:

(<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/>



The screenshot shows the VirtualBox.org website. At the top left is the VirtualBox logo, a blue cube with 'ORACLE' and 'VirtualBox' text. The main header features the 'VirtualBox' name in large blue letters. Below it is a 'Welcome to VirtualBox.org!' message. The left sidebar contains a list of links: About, Screenshots, Downloads, Documentation (with sub-links for End-user docs and Technical docs), Contribute, and Community. The main content area has three paragraphs: the first describes VirtualBox as a powerful x86 and AMD64/Intel64 virtualization product; the second lists supported guest operating systems; the third mentions frequent releases and community support. A large blue button in the center says 'Download VirtualBox 5.2'. On the right, a 'News Flash' box lists three recent releases: VirtualBox 5.2.16 (July 17th, 2018), VirtualBox 5.1.38 (May 9th, 2018), and VirtualBox 5.2 (October 18th, 2017). At the bottom, a 'Hot picks' section lists 'Pre-built virtual machines for developers at Oracle Tech Network'.

VirtualBox

search...
Login Preference

Welcome to VirtualBox.org!

VirtualBox is a powerful x86 and AMD64/Intel64 virtualization product for enterprise as well as home use. Not only is VirtualBox an extremely feature rich, high performance product for enterprise customers, it is also the only professional solution that is freely available as Open Source Software under the terms of the GNU General Public License (GPL) version 2. See "[About VirtualBox](#)" for an introduction.

Presently, VirtualBox runs on Windows, Linux, Macintosh, and Solaris hosts and supports a large number of [guest operating systems](#) including but not limited to Windows (NT 4.0, 2000, XP, Server 2003, Vista, Windows 7, Windows 8, Windows 10), DOS/Windows 3.x, Linux (2.4, 2.6, 3.x and 4.x), Solaris and OpenSolaris, OS/2, and OpenBSD.

VirtualBox is being actively developed with frequent releases and has an ever growing list of features, supported guest operating systems and platforms it runs on. VirtualBox is a community effort backed by a dedicated company: everyone is encouraged to contribute while Oracle ensures the product always meets professional quality criteria.

Download VirtualBox 5.2

News Flash

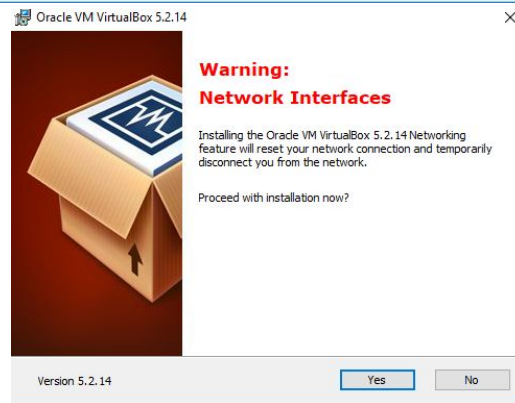
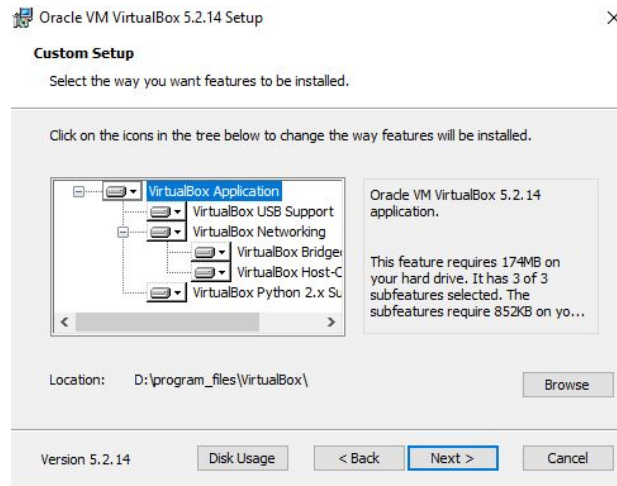
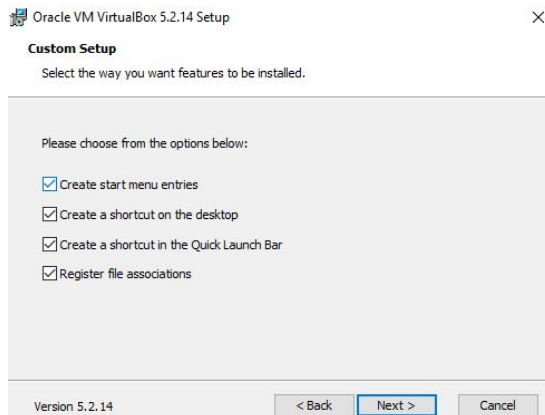
- **New July 17th, 2018**
VirtualBox 5.2.16 released!
Oracle today released a 5.2 maintenance release which improves stability and fixes regressions. See the [Changelog](#) for details.
- **New May 9th, 2018**
VirtualBox 5.1.38 released!
Oracle today released a 5.1 maintenance release which improves stability and fixes regressions. See the [Changelog](#) for details.
- **New October 18th, 2017**
VirtualBox 5.2 released!
Oracle today shipped a new minor release, VirtualBox 5.2. See the [announcement](#) for details.

[More information...](#)

Hot picks:

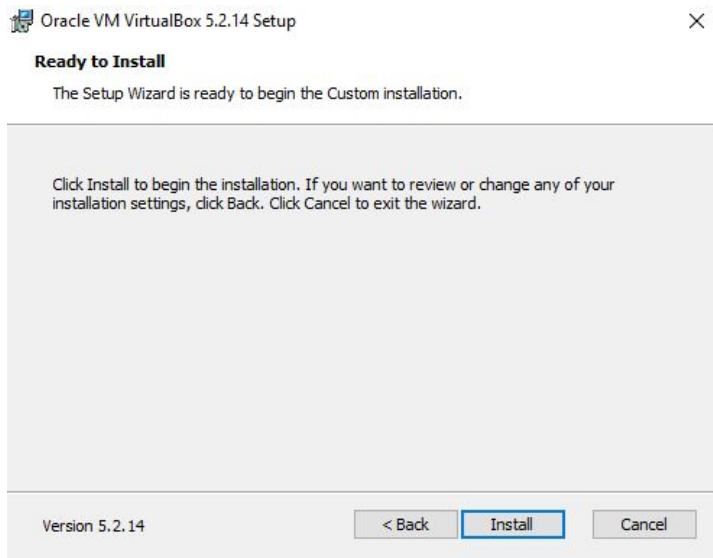
- Pre-built virtual machines for developers at [Oracle Tech Network](#)

NEXT->NEXT->NEXT->YES



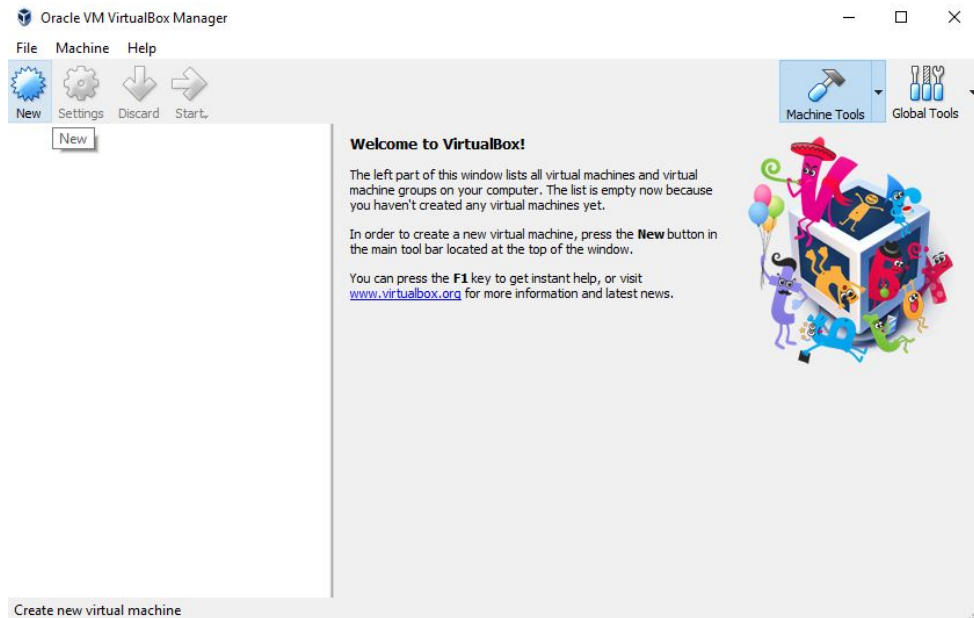
INSTALAÇÃO E CONFIGURAÇÃO DO AMBIENTE VIRTUAL

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



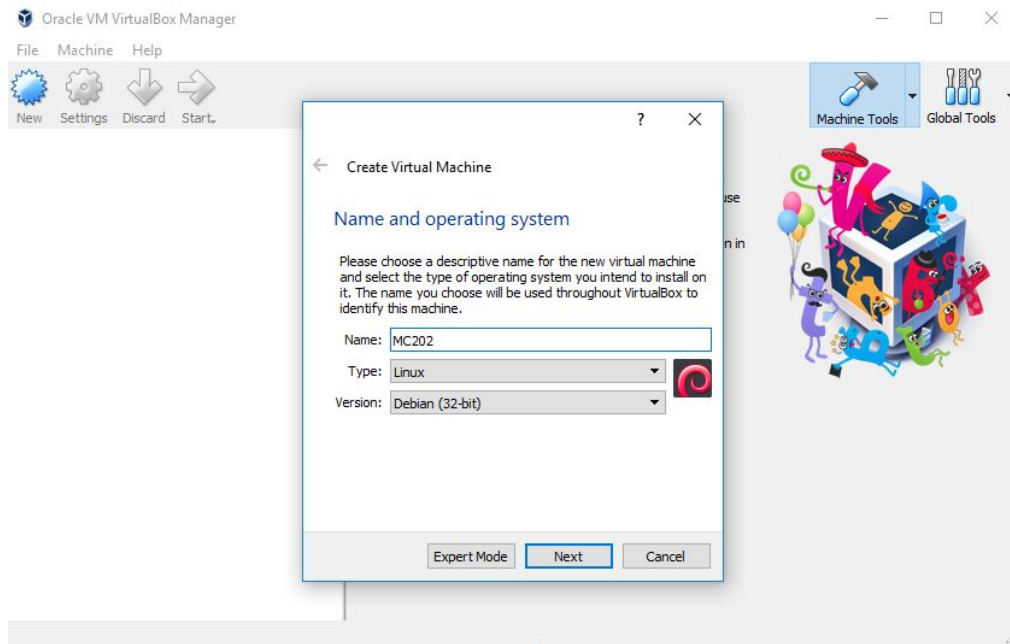
INSTALAÇÃO E CONFIGURAÇÃO DO AMBIENTE VIRTUAL

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



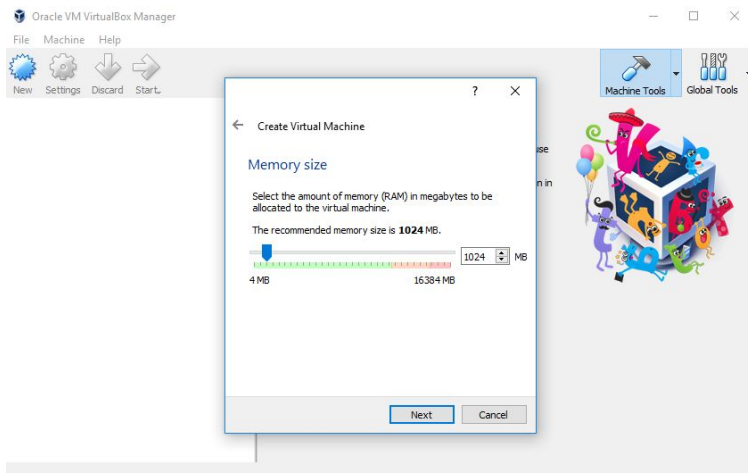
INSTALAÇÃO E CONFIGURAÇÃO DO AMBIENTE VIRTUAL

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



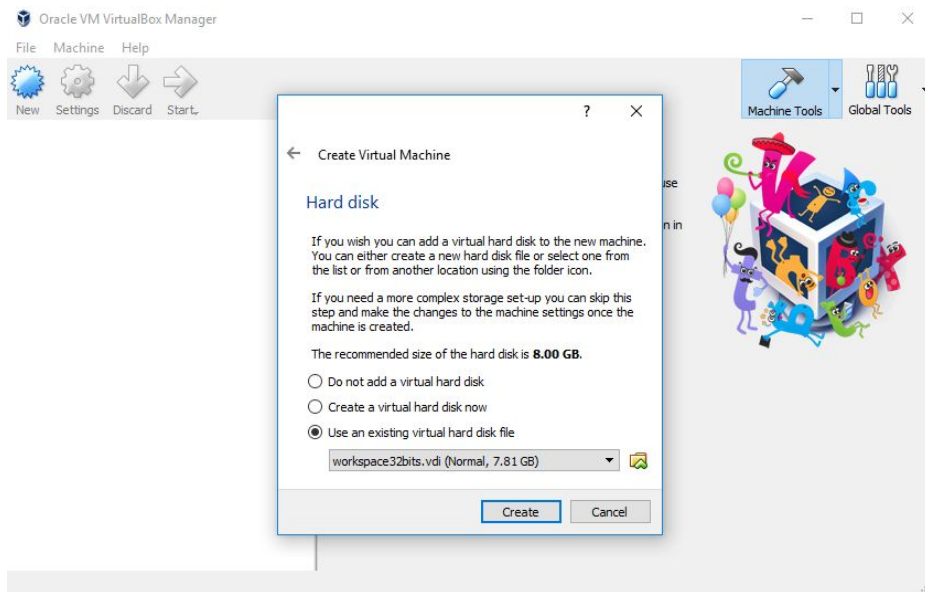
INSTALAÇÃO E CONFIGURAÇÃO DO AMBIENTE VIRTUAL

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)



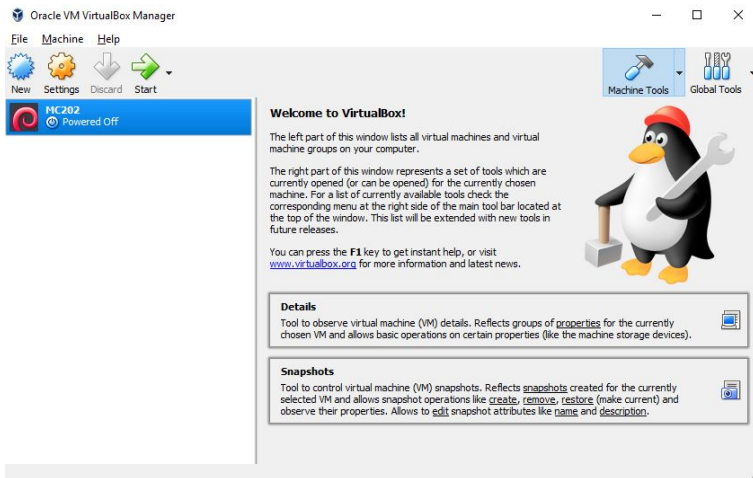
INSTALAÇÃO E CONFIGURAÇÃO DO AMBIENTE VIRTUAL

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

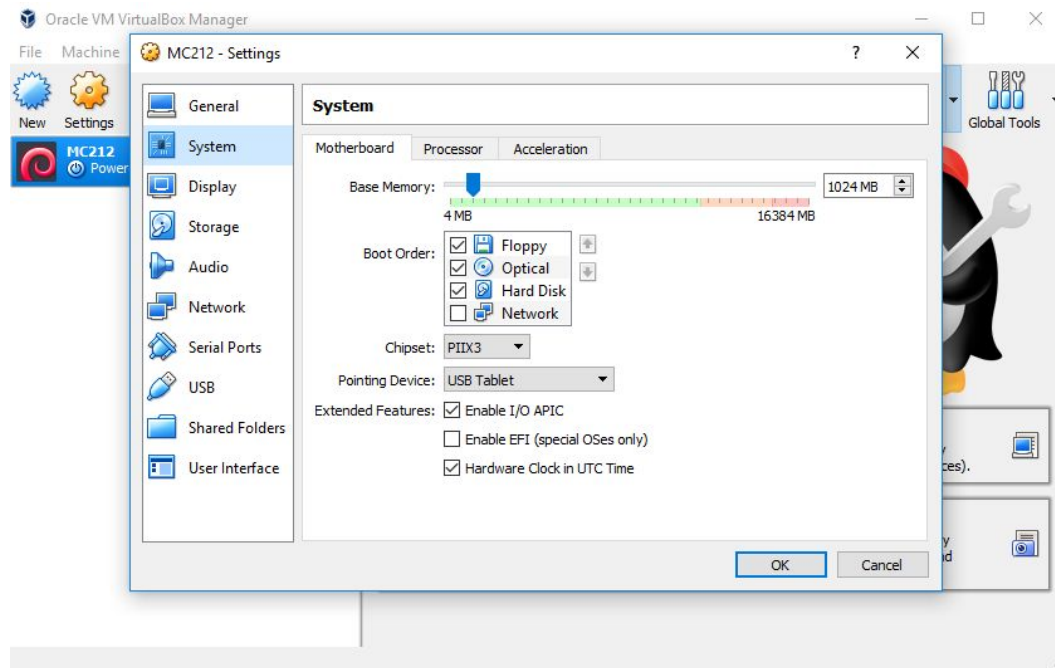
O ambiente virtual deveria estar pronto, clicando em “Start” ele deveria começar, mas ainda é possível que ele falhe, se isso acontecer, existem algumas configurações que ainda podemos arrumar.



INSTALAÇÃO E CONFIGURAÇÃO DO AMBIENTE VIRTUAL

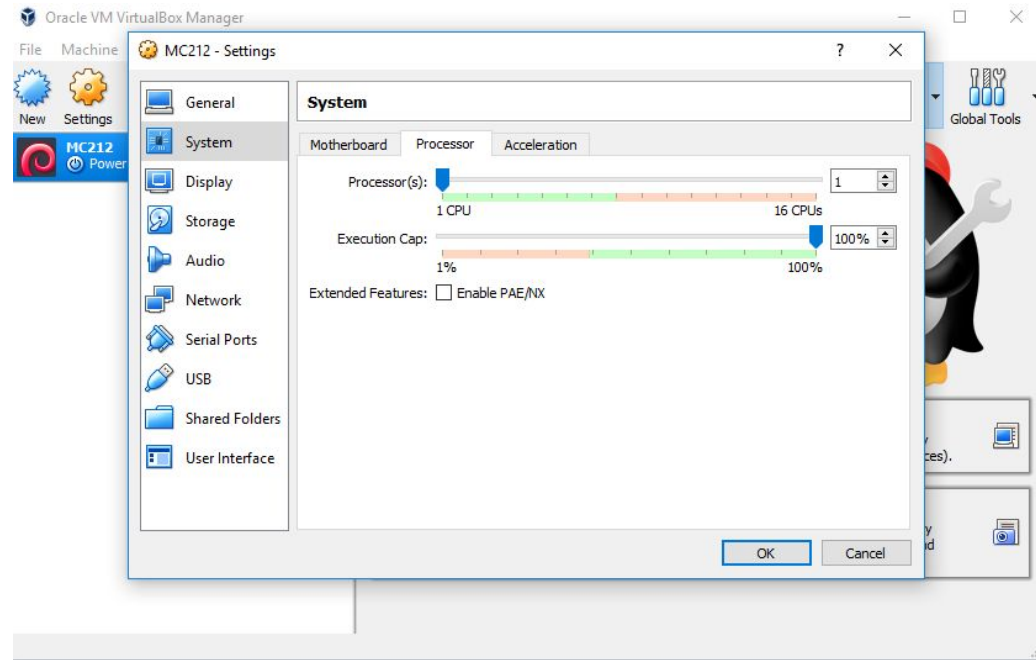
Em “Settings” escolha a aba “System” e habilite a opção “Enable I/O APIC”.

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.

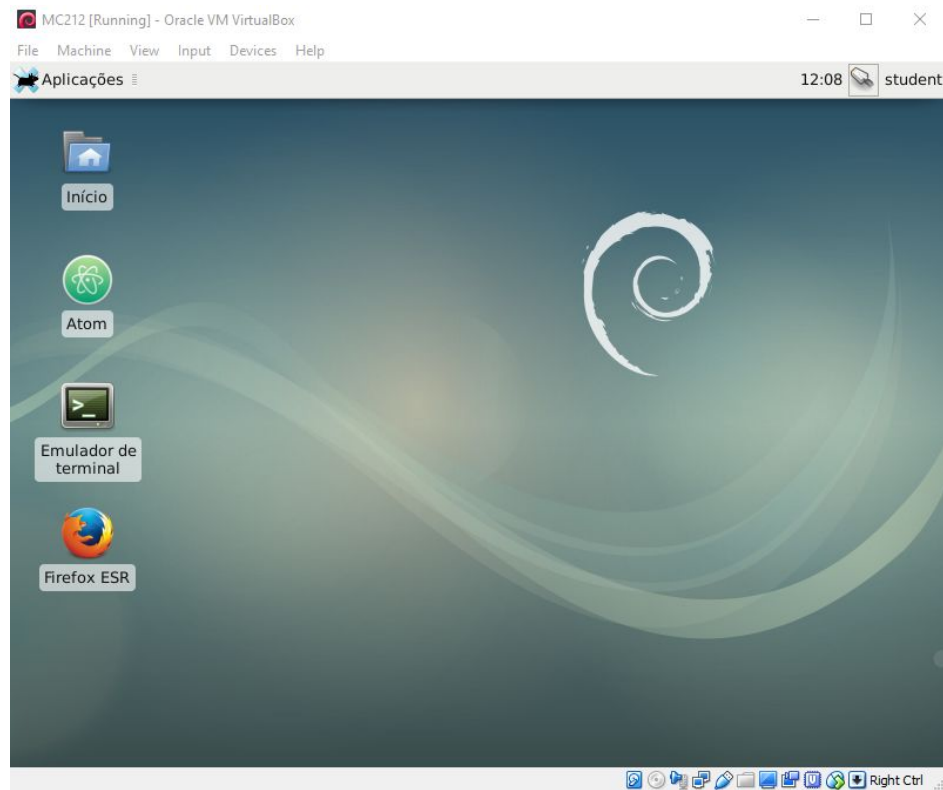


INSTALAÇÃO E CONFIGURAÇÃO DO AMBIENTE VIRTUAL

No ambiente você tem acesso ao “Firefox” para conseguir baixar os arquivos dos laboratórios.

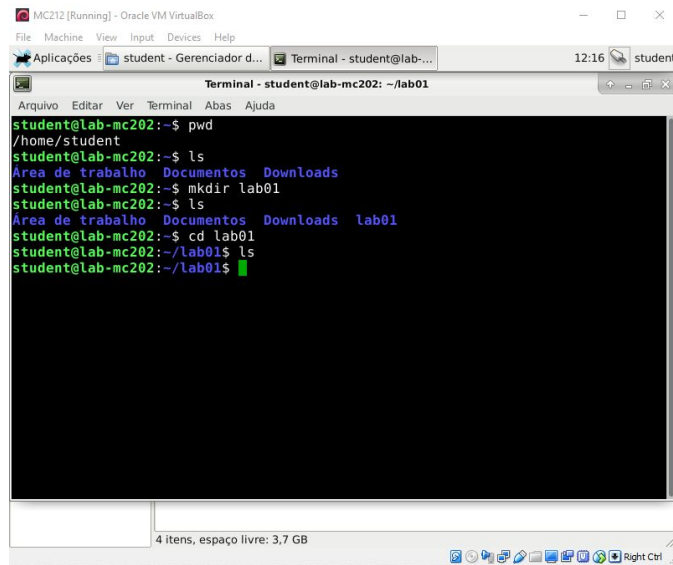
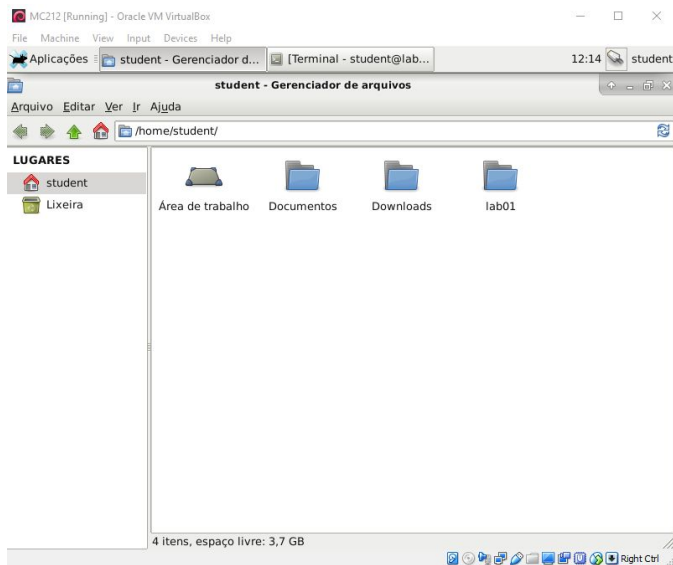
O editor “Atom” para escrever o código.

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

```
student@lab-mc202:~$ ls
Área de trabalho  Documentos  Downloads
student@lab-mc202:~$ pwd
/home/student
student@lab-mc202:~$ ls
Área de trabalho  Documentos  Downloads
student@lab-mc202:~$ mkdir lab01
student@lab-mc202:~$ ls
Área de trabalho  Documentos  Downloads  lab01
student@lab-mc202:~$ cd lab01/
student@lab-mc202:~/lab01$ pwd
/home/student/lab01
student@lab-mc202:~/lab01$ cd ..
student@lab-mc202:~$ pwd
/home/student
student@lab-mc202:~$
```

EMULADOR DE TERMINAL

Alguns comandos:

- **touch** <arquivo>- cria um arquivo vazio
- **cp** <arquivo> <copiar> - 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

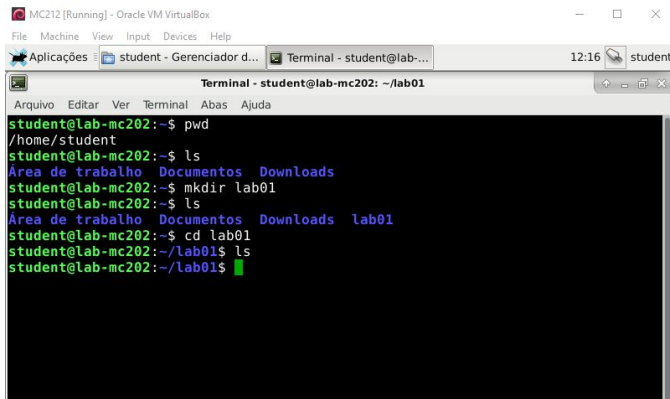
```
student@lab-mc202:~/lab01$ ls
student@lab-mc202:~/lab01$ touch lab00.c
student@lab-mc202:~/lab01$ ls
lab00.c
student@lab-mc202:~/lab01$ cp lab00.c lab01.c
student@lab-mc202:~/lab01$ ls
lab00.c  lab01.c
student@lab-mc202:~/lab01$ mv lab00.c ../
student@lab-mc202:~/lab01$ ls ../
Área de trabalho  Documentos  Downloads  lab00.c  lab01
student@lab-mc202:~/lab01$ cd ..
student@lab-mc202:~$ ls
Área de trabalho  Documentos  Downloads  lab00.c  lab01
student@lab-mc202:~$ rm lab00.c
student@lab-mc202:~$ ls
Área de trabalho  Documentos  Downloads  lab01
student@lab-mc202:~$ rm -rf lab01/
student@lab-mc202:~$ ls
Área de trabalho  Documentos  Downloads
student@lab-mc202:~$ █
```

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.

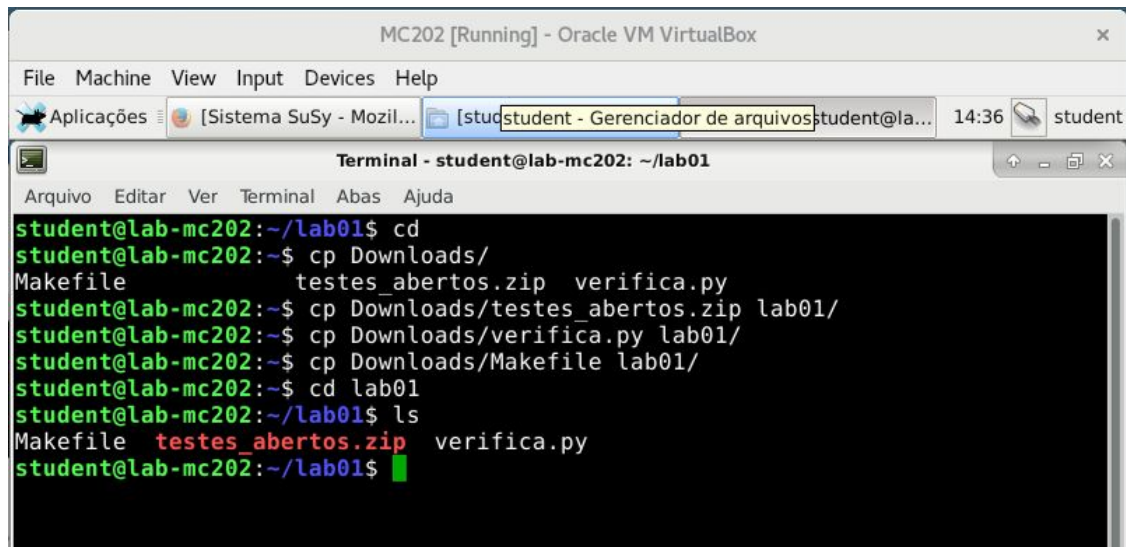


```
MC212 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Aplicações student - Gerenciador d... Terminal - student@lab-... 12:16 student
Terminal - student@lab-mc202: ~/lab01
Arquivo Editar Ver Terminal Abas Ajuda
student@lab-mc202:~$ pwd
/home/student
student@lab-mc202:~$ ls
Área de trabalho Documentos Downloads
student@lab-mc202:~$ mkdir lab01
student@lab-mc202:~$ ls
Área de trabalho Documentos Downloads lab01
student@lab-mc202:~$ cd lab01
student@lab-mc202:~/lab01$ ls
student@lab-mc202:~/lab01$
```

EXEMPLO: LAB01

Os arquivos baixados vão estar no diretório:
/home/student/Downloads.

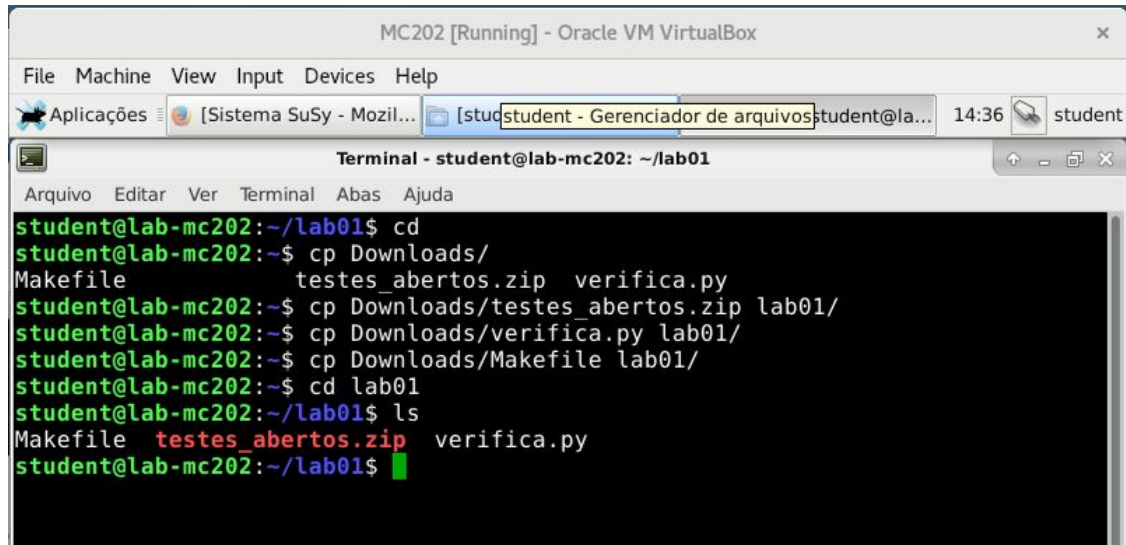
Vamos mover os arquivos agora para a pasta "lab01" com os seguintes comandos:

A screenshot of a terminal window titled "Terminal - student@lab-mc202: ~/lab01" within an Oracle VM VirtualBox environment. The terminal shows a series of commands being executed to move files from the Downloads directory to the lab01 directory. The files being moved are Makefile, testes_abertos.zip, and verifica.py. The terminal output shows the current directory, the files being copied, and the final directory listing showing the files successfully moved.

```
MC202 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Aplicações [Sistema SuSy - Mozil... [studstudent - Gerenciador de arquivos]student@la... 14:36 student
Terminal - student@lab-mc202: ~/lab01
Arquivo Editar Ver Terminal Abas Ajuda
student@lab-mc202:~/lab01$ cd
student@lab-mc202:~$ cp Downloads/
Makefile testes_abertos.zip verifica.py
student@lab-mc202:~$ cp Downloads/testes_abertos.zip lab01/
student@lab-mc202:~$ cp Downloads/verifica.py lab01/
student@lab-mc202:~$ cp Downloads/Makefile lab01/
student@lab-mc202:~$ cd lab01
student@lab-mc202:~/lab01$ ls
Makefile testes_abertos.zip verifica.py
student@lab-mc202:~/lab01$
```

EXEMPLO: LAB01

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

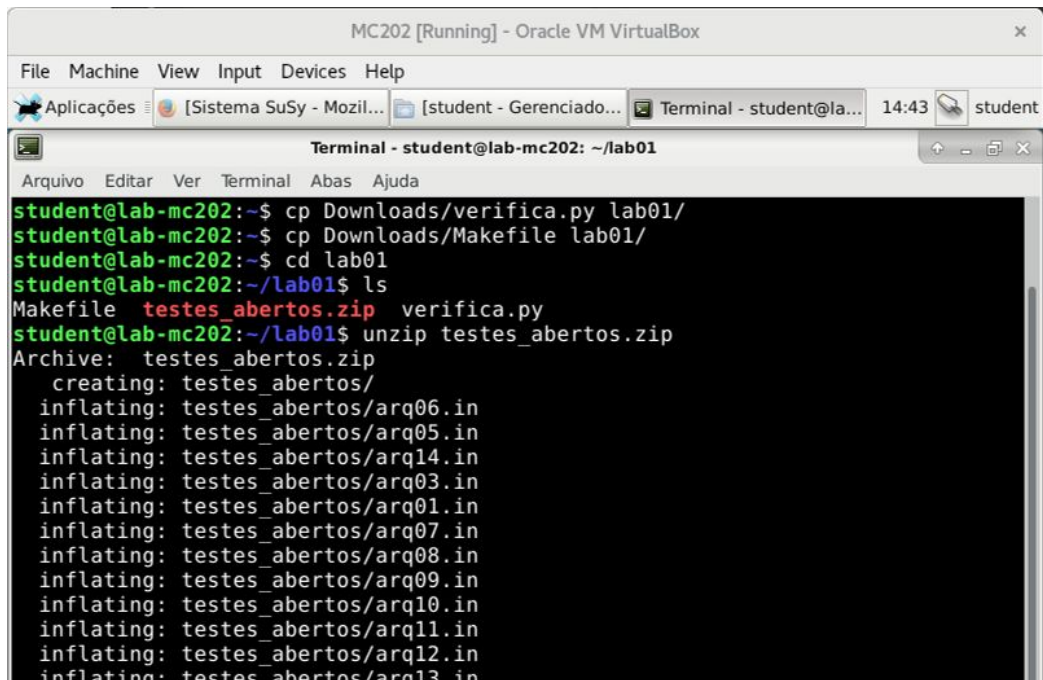


The screenshot shows a terminal window titled "Terminal - student@lab-mc202: ~/lab01" within an Oracle VM VirtualBox environment. The terminal displays the following commands and their outputs:

```
student@lab-mc202:~/lab01$ cd
student@lab-mc202:~$ cp Downloads/
Makefile      testes_abertos.zip  verifica.py
student@lab-mc202:~$ cp Downloads/testes_abertos.zip lab01/
student@lab-mc202:~$ cp Downloads/verifica.py lab01/
student@lab-mc202:~$ cp Downloads/Makefile lab01/
student@lab-mc202:~$ cd lab01
student@lab-mc202:~/lab01$ ls
Makefile  testes_abertos.zip  verifica.py
student@lab-mc202:~/lab01$
```

EXEMPLO: LAB01

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



The screenshot shows a terminal window titled "Terminal - student@lab-mc202: ~/lab01" within an Oracle VM VirtualBox environment. The terminal displays the following commands and output:

```
student@lab-mc202:~$ cp Downloads/verifica.py lab01/
student@lab-mc202:~$ cp Downloads/Makefile lab01/
student@lab-mc202:~$ cd lab01
student@lab-mc202:~/lab01$ ls
Makefile  testes_abertos.zip  verifica.py
student@lab-mc202:~/lab01$ unzip testes_abertos.zip
Archive:  testes_abertos.zip
  creating: testes_abertos/
  inflating: testes_abertos/arq06.in
  inflating: testes_abertos/arq05.in
  inflating: testes_abertos/arq14.in
  inflating: testes_abertos/arq03.in
  inflating: testes_abertos/arq01.in
  inflating: testes_abertos/arq07.in
  inflating: testes_abertos/arq08.in
  inflating: testes_abertos/arq09.in
  inflating: testes_abertos/arq10.in
  inflating: testes_abertos/arq11.in
  inflating: testes_abertos/arq12.in
  inflating: testes_abertos/arq13.in
```

EXEMPLO: LAB01

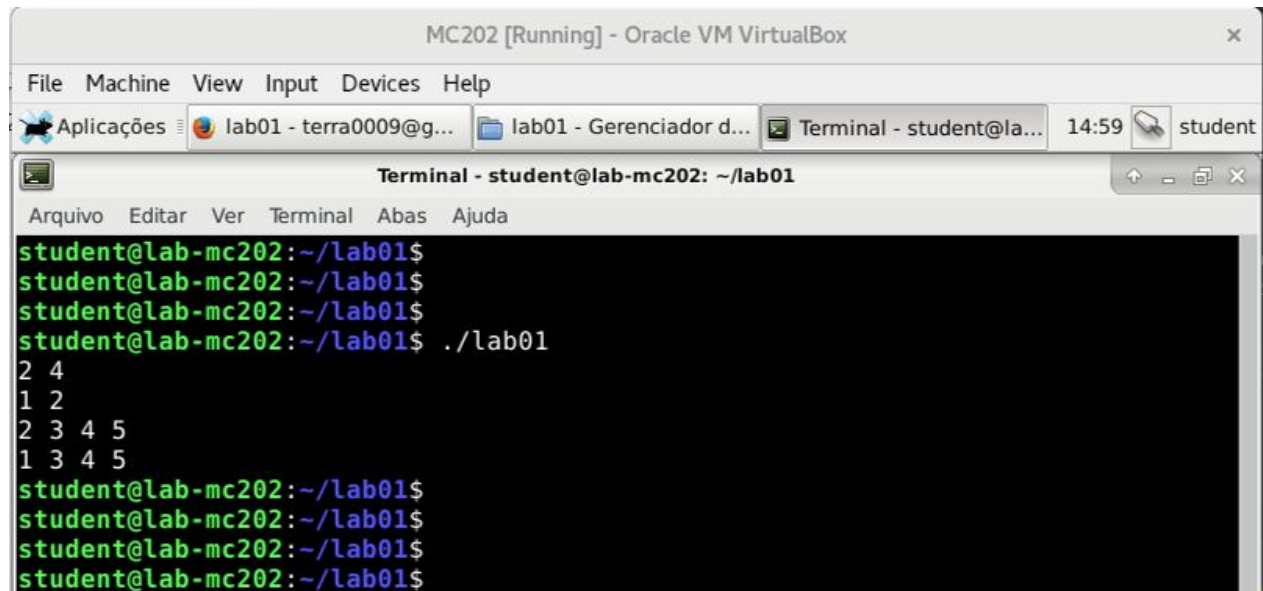
make lab01 - O aluno deve criar o arquivo lab01.c que vai conter o código do laboratório. O comando make lab01 compila o código do lab01.c criando o executável "lab01"

[illegible]

EXEMPLO: LAB01

`./lab01` - Esse comando executa o programa `lab01`.

Nesse caso, a entrada do programa deve ser especificada pelo próprio usuário como mostrado abaixo.

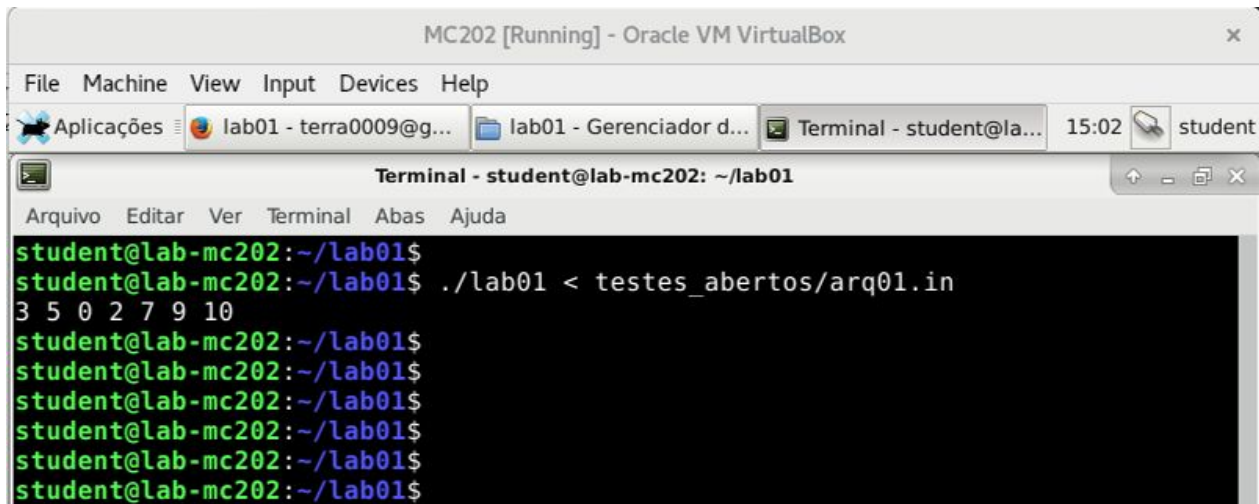


The screenshot shows a VirtualBox window titled "MC202 [Running] - Oracle VM VirtualBox". Inside, there is a terminal window titled "Terminal - student@lab-mc202: ~/lab01". The terminal shows the following sequence of commands and output:

```
student@lab-mc202:~/lab01$  
student@lab-mc202:~/lab01$  
student@lab-mc202:~/lab01$  
student@lab-mc202:~/lab01$ ./lab01  
2 4  
1 2  
2 3 4 5  
1 3 4 5  
student@lab-mc202:~/lab01$  
student@lab-mc202:~/lab01$  
student@lab-mc202:~/lab01$  
student@lab-mc202:~/lab01$
```

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"



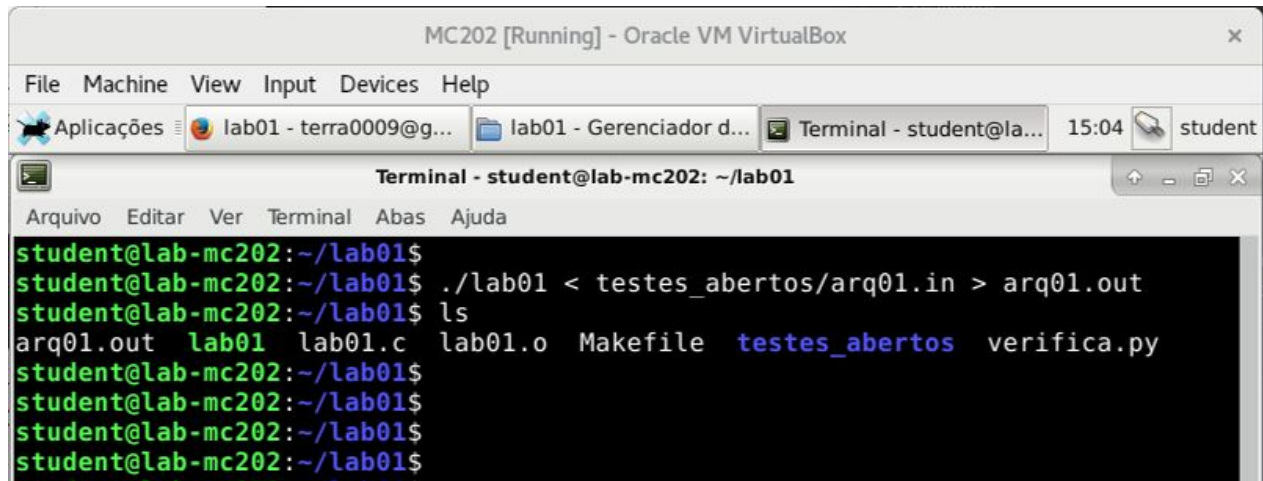
The screenshot shows a terminal window titled "Terminal - student@lab-mc202: ~/lab01" within an Oracle VM VirtualBox environment. The terminal displays the following sequence of commands and output:

```
student@lab-mc202:~/lab01$  
student@lab-mc202:~/lab01$ ./lab01 < testes_abertos/arq01.in  
3 5 0 2 7 9 10  
student@lab-mc202:~/lab01$  
student@lab-mc202:~/lab01$  
student@lab-mc202:~/lab01$  
student@lab-mc202:~/lab01$  
student@lab-mc202:~/lab01$  
student@lab-mc202:~/lab01$  
student@lab-mc202:~/lab01$
```

The output of the program is the sequence of numbers "3 5 0 2 7 9 10". The terminal window also shows the standard Ubuntu desktop environment with various application icons in the top bar.

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.



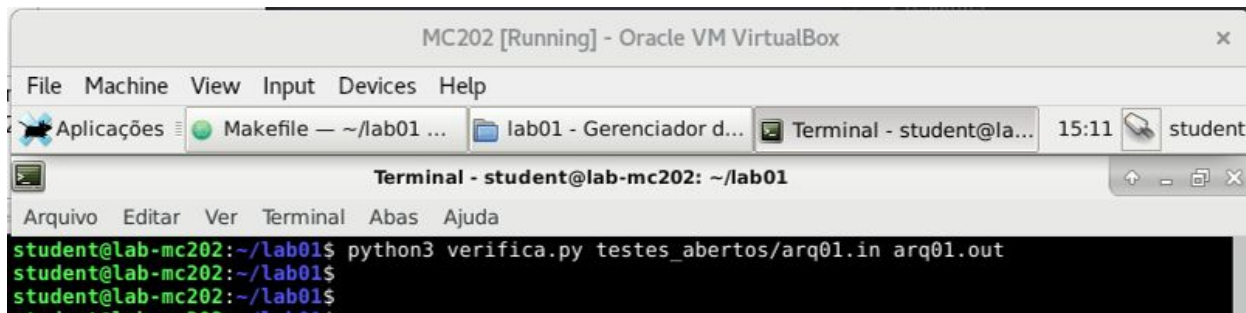
```
MC202 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Aplicações lab01 - terra0009@g... lab01 - Gerenciador d... Terminal - student@la... 15:04 student
Terminal - student@lab-mc202: ~/lab01
Arquivo Editar Ver Terminal Abas Ajuda
student@lab-mc202:~/lab01$
student@lab-mc202:~/lab01$ ./lab01 < testes_abertos/arq01.in > arq01.out
student@lab-mc202:~/lab01$ ls
arq01.out lab01 lab01.c lab01.o Makefile testes_abertos verifica.py
student@lab-mc202:~/lab01$
student@lab-mc202:~/lab01$
student@lab-mc202:~/lab01$
student@lab-mc202:~/lab01$
```

EXEMPLO: LAB01

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

O código em python "verifica.py" é responsável 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.



The screenshot shows a VirtualBox window titled "MC202 [Running] - Oracle VM VirtualBox". Inside, there is a terminal window titled "Terminal - student@lab-mc202: ~/lab01". The terminal shows the following commands and output:

```
student@lab-mc202:~/lab01$ python3 verifica.py testes_abertos/arq01.in arq01.out
student@lab-mc202:~/lab01$
student@lab-mc202:~/lab01$
```

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?