

Universidade Federal De Uberlândia

Faculdade de Computação

Sistemas Operacionais

GSI018

EULLER HENRIQUE BANDEIRA OLIVEIRA

11821BSI210

Relatório :

Atividade Prática de Fixação

(Unidade I)

Uberlândia

2021

# Sumário

Questão 1.....	1
Conceitos:.....	1
Identificações:.....	2
Questão 2.....	4
Questão 3.....	5
Questão 4.....	11
Questão 5.....	15
Questão 6.....	17
Questão 7.....	22
Questão 8.....	24
Questão 9.....	26
Questão 10.....	29
Código:.....	29
Descrição:.....	37
Main:.....	37
Executar:.....	38
ObterDataHora:.....	39
CLS:.....	40
DIRENT:.....	40
DATE:.....	42
TIME:.....	42
Compilação e execução.....	43
Compilação do SHELL:.....	43
Execução do SHELL.....	43
Execução do comando CLS:.....	43
Execução do comando DIR:.....	44
Execução do comando TIME:.....	44
Execução do comando DATE:.....	44

**Computadores utilizados:**

Computador 1: Notebook Samsung NP300E4L-KW1BR, PROCESSADOR: i3 (sexta geração) , RAM: 4gb, HD: 1tb, SO: Linux Mint 19.3 Cinnamon

Computador 2: Notebook Samsung NP2E705G-KD1BR, PROCESSADOR: i3 (terceira geração), RAM: 4gb, HD: 500gb, SO: Windows 10 Home

# Questão 1

**Ao ligar o computador, busque identificar as etapas de execução do: POST, Boot Manager, Boot Loader e inicialização do Sistema operacional.**

## Conceitos:

POST:

O processador executa a rotina POST (essa rotina faz parte do firmware, ou seja, ela foi armazenada na memória ROM). Tal rotina possui a função de testar todos os dispositivos do computador, esse teste serve para verificar se o hardware está apto para rodar o software, ou seja, o SO.

Boot Manager:

No processo de iniciação do computador, a rotina boot manager será executada caso o computador possua mais de um sistema operacional. Tal rotina possui a função de exibir para o usuário uma interface gráfica com as opções de SO disponíveis. Após a exibição, a rotina ficará no aguardo pela resposta do usuário por meio do teclado ou do mouse.

Boot Loader:

Já o boot loader é uma rotina que possui todo o conjunto de instruções necessárias para carregar um determinado SO, ou seja, cada SO possui seu próprio boot loader, sem o boot loader o computador não saberia como começar a executar determinado SO, pois cada SO foi desenvolvido de uma forma diferente, não existe um padrão, logo, o boot loader é extremamente importante.

Kernel:

O kernel, ou seja, o núcleo do SO, é executado. O kernel possui a função de carregar as informações básicas do sistema operacional, os principais arquivos e relacionar os componentes de hardware com os respectivos drivers.

Inicialização do SO:

Nesta última etapa, a interface gráfica do SO é exibida, caso o computador possua senha a tela de login será exibida, caso contrário a tela com a área de trabalho será exibida. Desta maneira, o usuário conseguirá utilizar o SO em questão.

## **Identificações:**

Computador utilizado: computador 1.

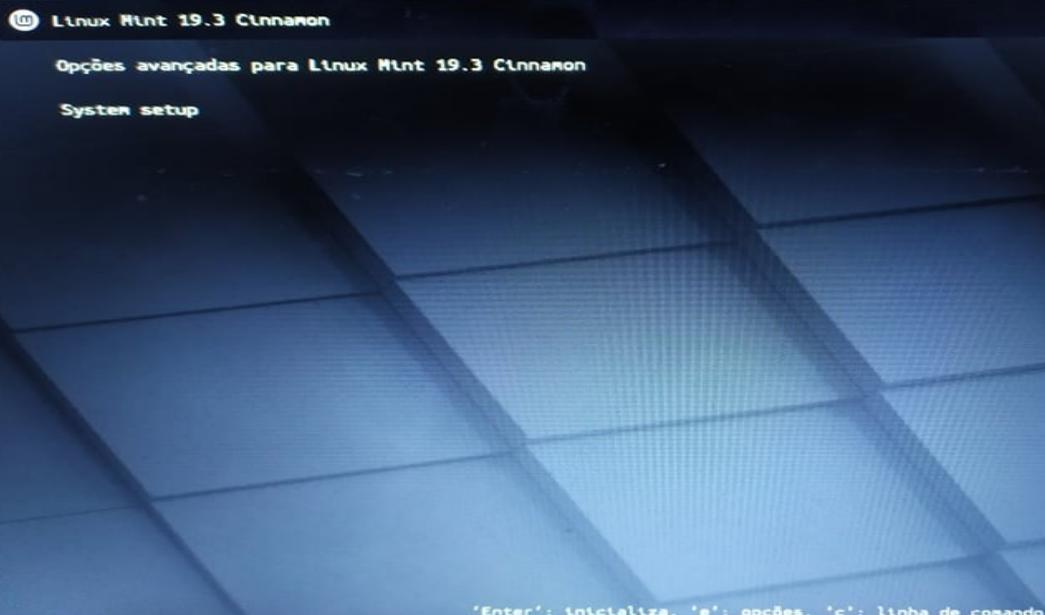
POST: Se o hardware não fosse apto para rodar o SO, bips específicos seriam emitidos indicando o que está causando essa incompatibilidade. Como o computador 1 não emitiu nenhum bip, a rotina POST indicou que o computador estava apto para rodar o SO, sendo assim, a sequência de boot continuou normalmente.

Quando a rotina post testa a tela, a logo da fabricante é mostrada.

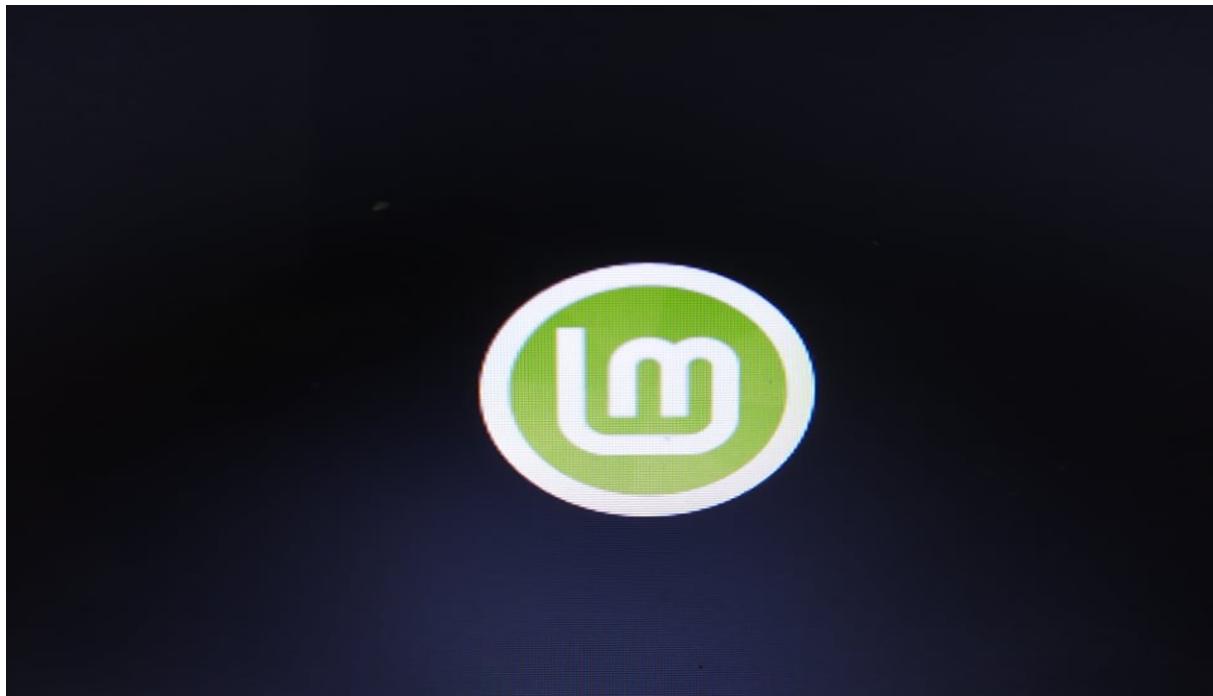


Boot Manager: O computador 1 não possui dois sistemas operacionais, logo, normalmente, essa rotina não é executada no processo de boot do computador.

Se o computador 1 possuísse mais de um sistema operacional, o boot manager seria assim:



Boot Loader: A logo do linux mint aparece na tela. Após isso ocorrer o kernel começa a ser carregado, a indicação de que isso ocorre é visível na logo presente na tela (pois o plano de fundo da logo do SO gira) e no tempo de espera.



Inicialização do Sistema Operacional: A interface gráfica do linux mint é exibida. Se o computador tivesse sido configurado para requisitar a senha quando ele fosse ligado, a tela de login seria exibida, como essa configuração não ocorreu, a área de trabalho foi exibida.



## Questão 2

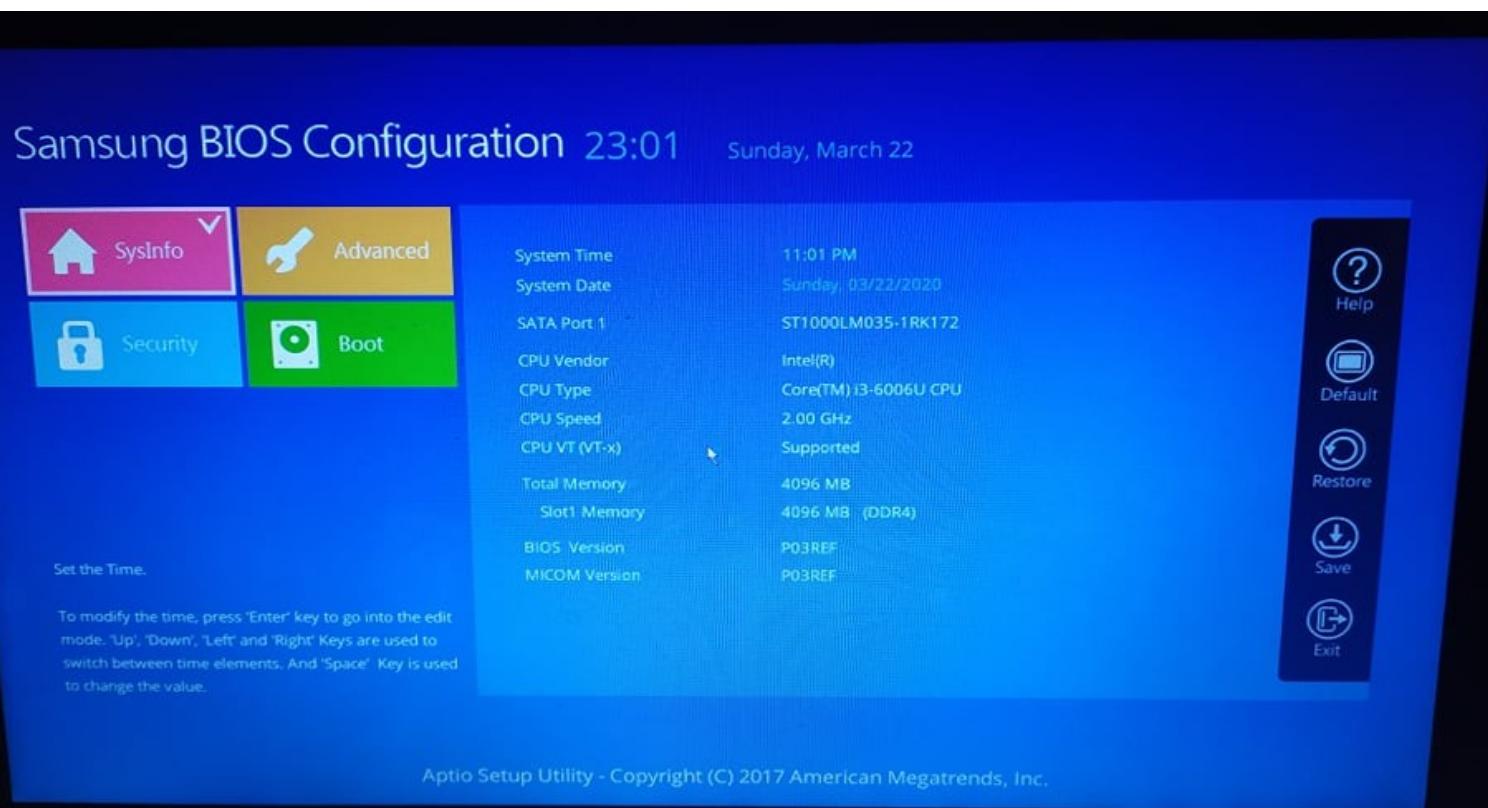
**Ao ligar o computador, busque identificar o fabricante e a data do firmware do seu computador.**

Computador utilizado: computador 1.

Essas informações não aparecem ao ligar o computador. Para descobrir a fabricante do firmware do computador é preciso acessar o SETUP do BIOS do computador. Para acessar o SETUP do BIOS é preciso apertar a tecla f10 ao ligar o computador.

Fabricante do firmware: American Megatrends Inc.

Data do firmaware: 2017



É possível descobrir a data exata após o SO ser inicializado por meio do comando “sudo dmidecode -s bios-release-date”.

```
euller@euller:~$ sudo dmidecode -s bios-release-date
12/21/2017
```

## Questão 3

Ligue o computador e acesse o firmware SETUP. Navegue nas opções disponíveis e se familiarize com elas. Faça isso em computadores diferentes para ver as diferenças de funcionalidades entre o firmware de diferentes fabricantes.

Diferenças entre o firmware SETUP do computador 1 e do computador 2:

1. Interface gráfica

Enquanto a interface gráfica do computador 1 é moderna, a interface gráfica do computador 2 é antiquada.

2. Algumas informações e funções estão presentes somente no computador 1

Ex: A função CPU POWER SAVING MODE está presente somente no computador 1

3. Algumas informações e funções estão presentes somente no computador 2

Ex: A função TPM Configuration está presente somente no computador 2

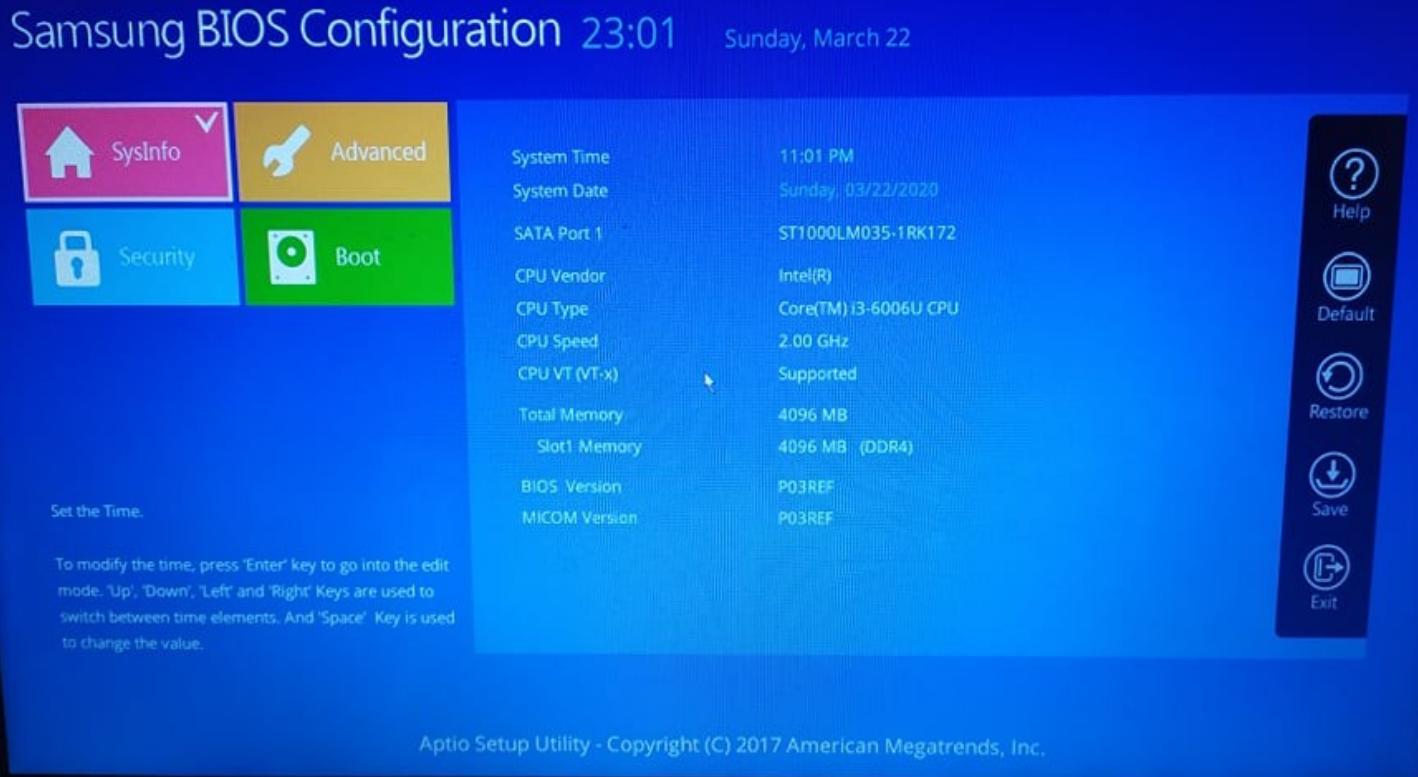
4. Algumas informações e funções estão presentes em menus diferentes

Ex: A função Touch Pad Mouse está localizada no menu Advanced do computador 1, já no computador 2 ela está localizada no menu Boot

5. Fabricantes

Enquanto a fabricante do firmware SETUP no computador 1 é a American Megatrends, Inc, a fabricante no computador 2 é a Phoenix.

No computador 1:



## Samsung BIOS Configuration 23:01 Sunday, March 22



# Samsung BIOS Configuration 23:01 Sunday, March 22

The screenshot shows the Samsung BIOS Configuration interface. On the left, there's a navigation menu with four main options: SysInfo (pink), Advanced (yellow), Security (blue, currently selected), and Boot (green). On the right, under the Security section, there are password-related settings:

- Supervisor Password: Clear
- User Password: Clear
- HDD Password: Clear
- Set Supervisor Password > (link)
- Set User Password > (link)
- Set HDD Password > (link)
- >Password On Boot: On (with a slider bar)
- Secure Boot Configuration > (link)
- TPM Configuration > (link)

A note below the menu states: "Supervisor Password controls access to the setup utility."

On the far right, a vertical toolbar provides quick access to Help, Default, Restore, Save, and Exit.

Aptio Setup Utility - Copyright (C) 2017 American Megatrends, Inc.

# Samsung BIOS Configuration 23:01 Sunday, March 22

This screenshot shows the Boot configuration page of the Samsung BIOS. The left sidebar has the same four tabs: SysInfo, Advanced, Security, and Boot (selected). The main pane displays boot-related settings:

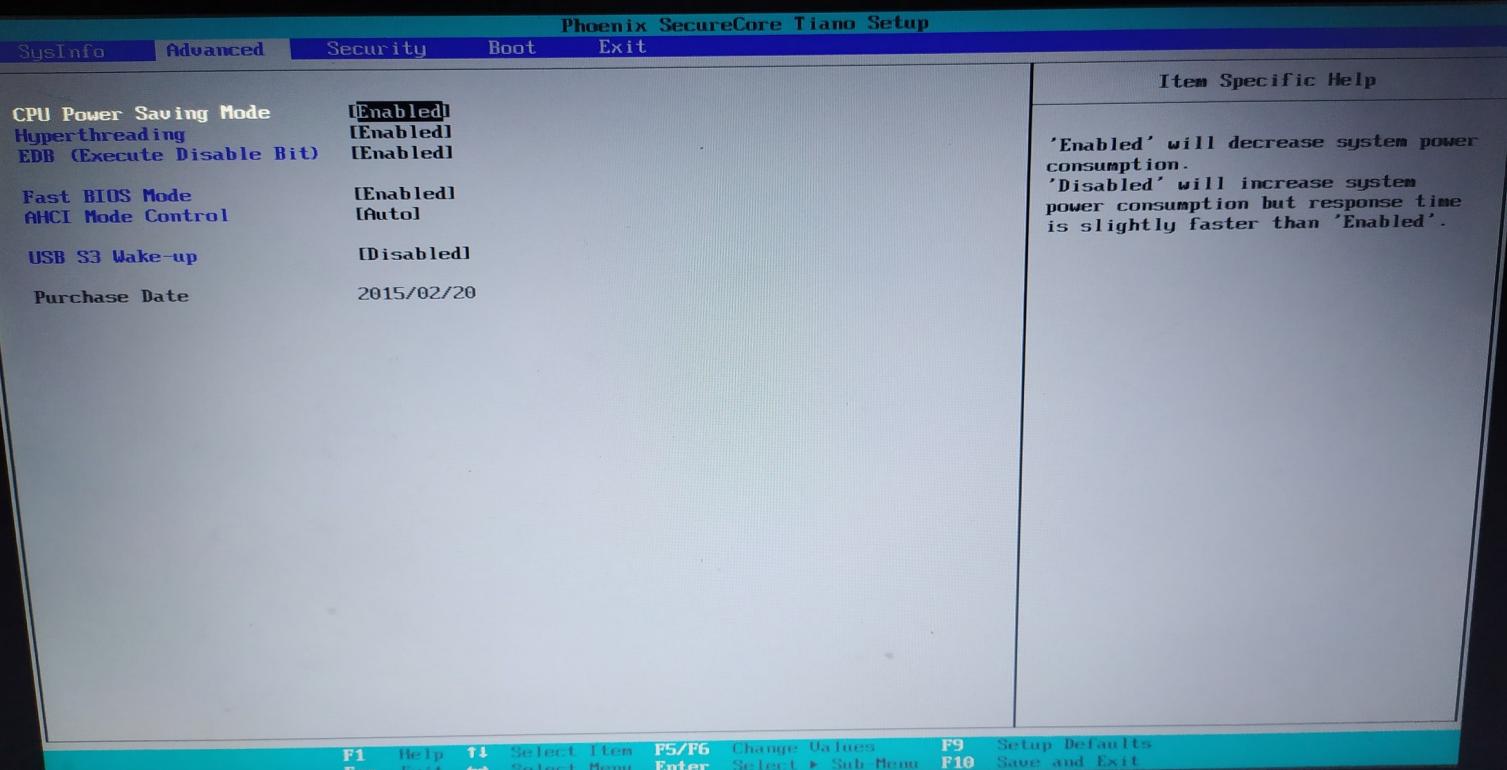
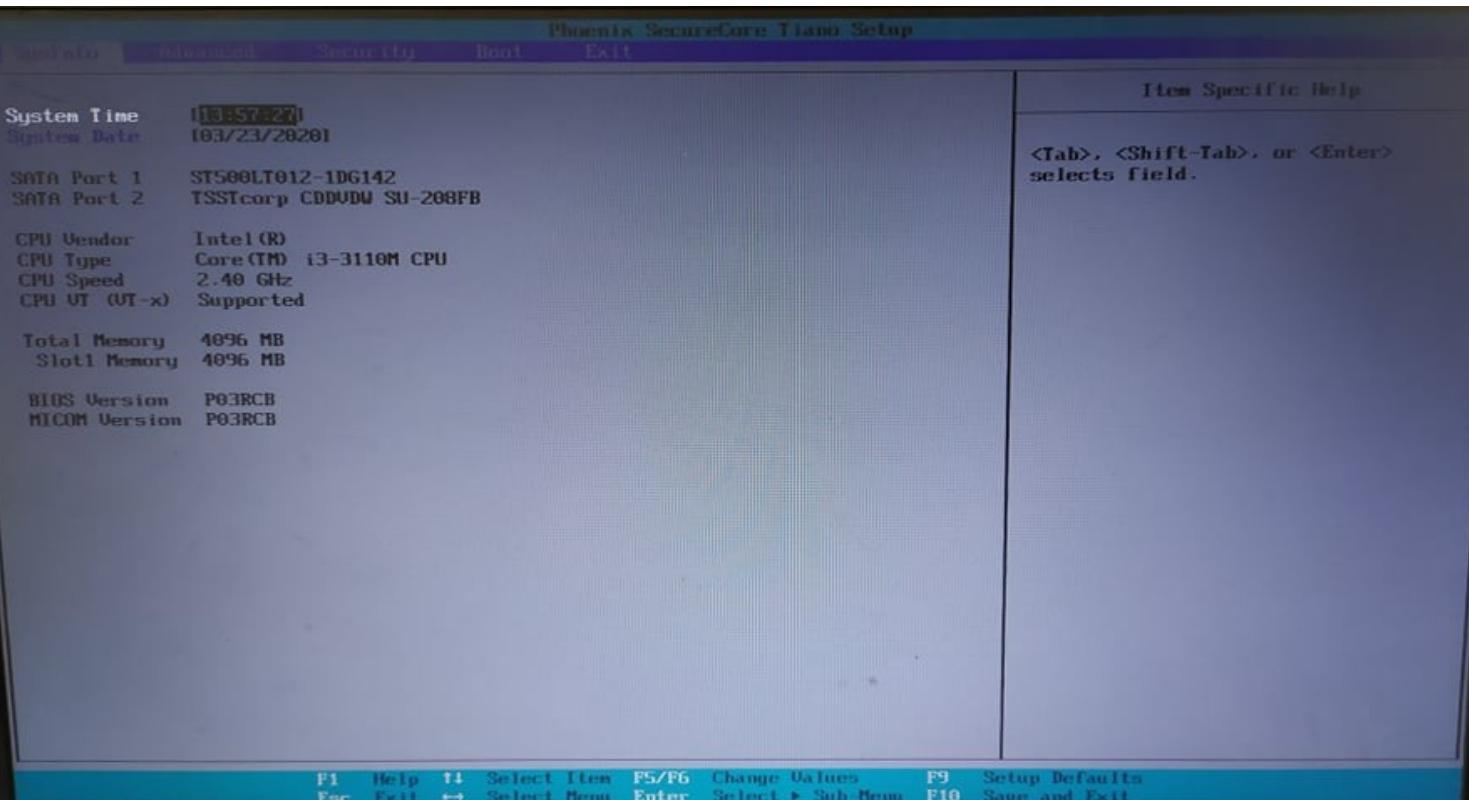
- Boot Device Priority > (link)
- Secure Boot Control: On (with a slider bar)
- Fast BIOS Mode: On (with a slider bar)
- PXE OPROM: Off (with a slider bar)

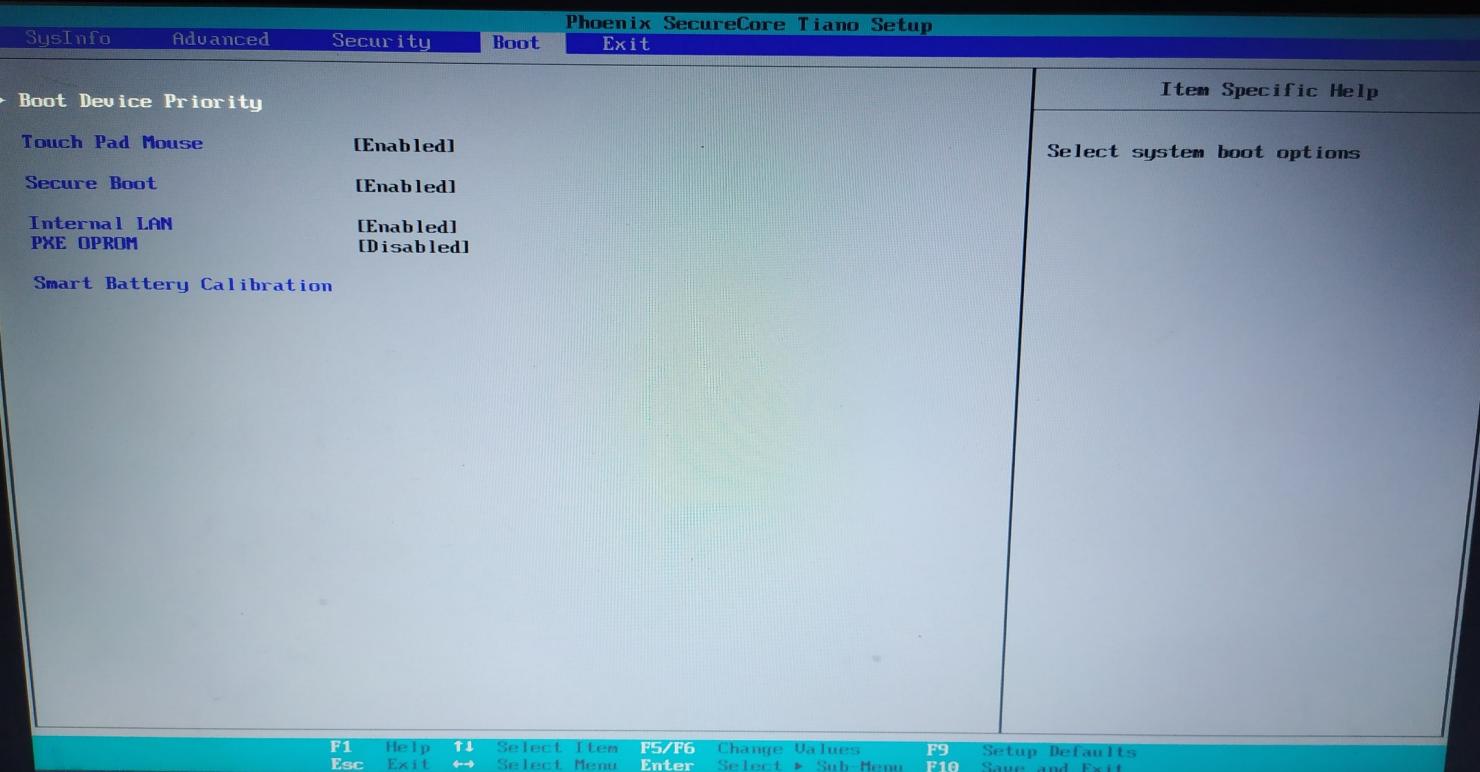
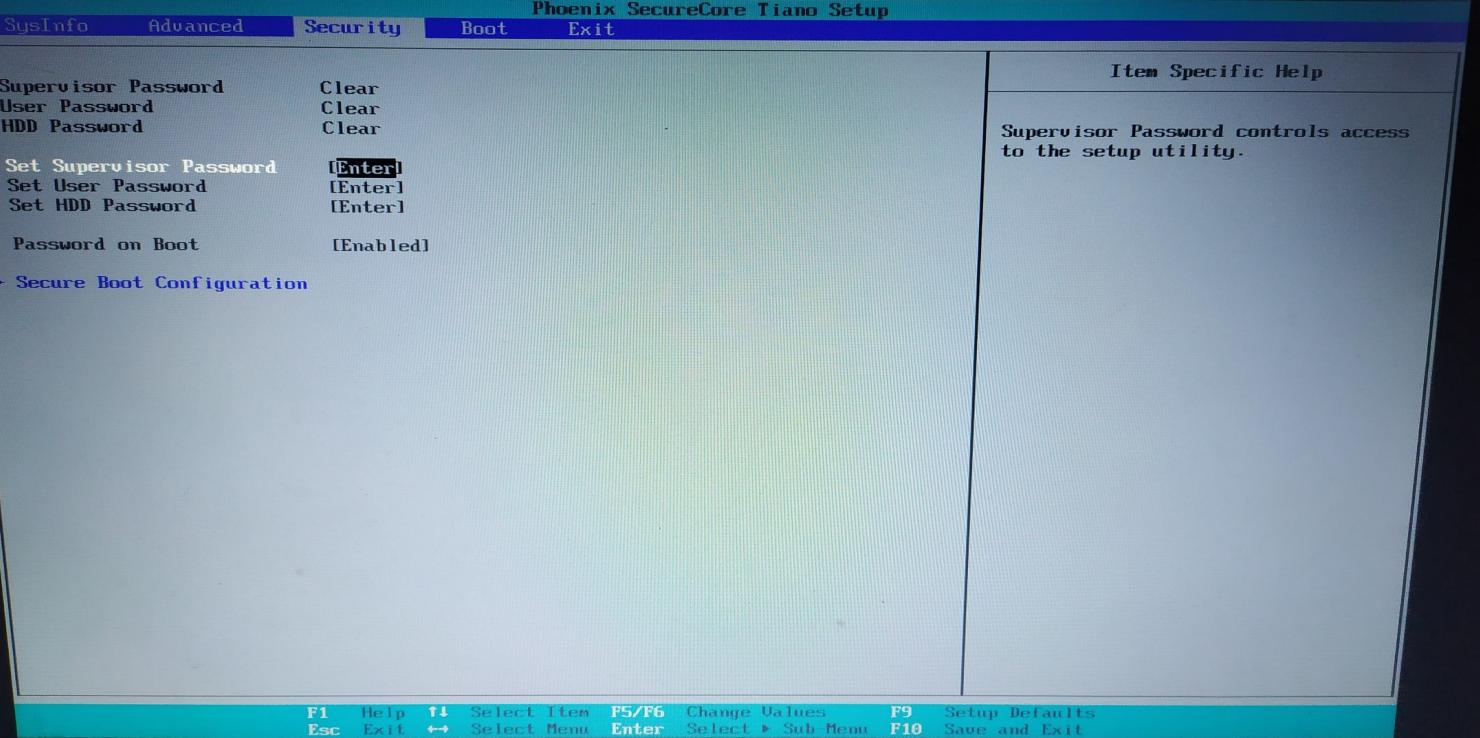
A warning message in the center states: "WARNING!! If you select incorrect option, system may not boot to OS correctly."

At the bottom left, it says "Select system boot options." A note at the bottom right indicates: "Aptio Setup Utility - Copyright (C) 2017 American Megatrends, Inc."

On the far right, a vertical toolbar provides quick access to Help, Default, Restore, Save, and Exit.

No computador 2:





## Phoenix SecureCore Tiano Setup

SysInfo Advanced Security Boot

Exit

Exit Saving Changes  
Exit Discarding Changes  
Load Setup Defaults  
Discard Changes  
Save Changes

Item Specific Help

Exit Setup and save your changes.

F1 Help ↑ Select Item F5/F6 Change Values F9 Setup Defaults  
Esc Exit ↔ Select Menu Enter Select ▶ Sub-Menu F10 Save and Exit

## Questão 4

Ligue o computador e identifique a configuração (marca, modelo, capacidade, quantidade instalada) de processadores, memória principal e memória secundária. Para isso, acesse as informações do firmware de SETUP e utilize ferramentas que acompanham o sistema operacional.

Computador utilizado: computador 1

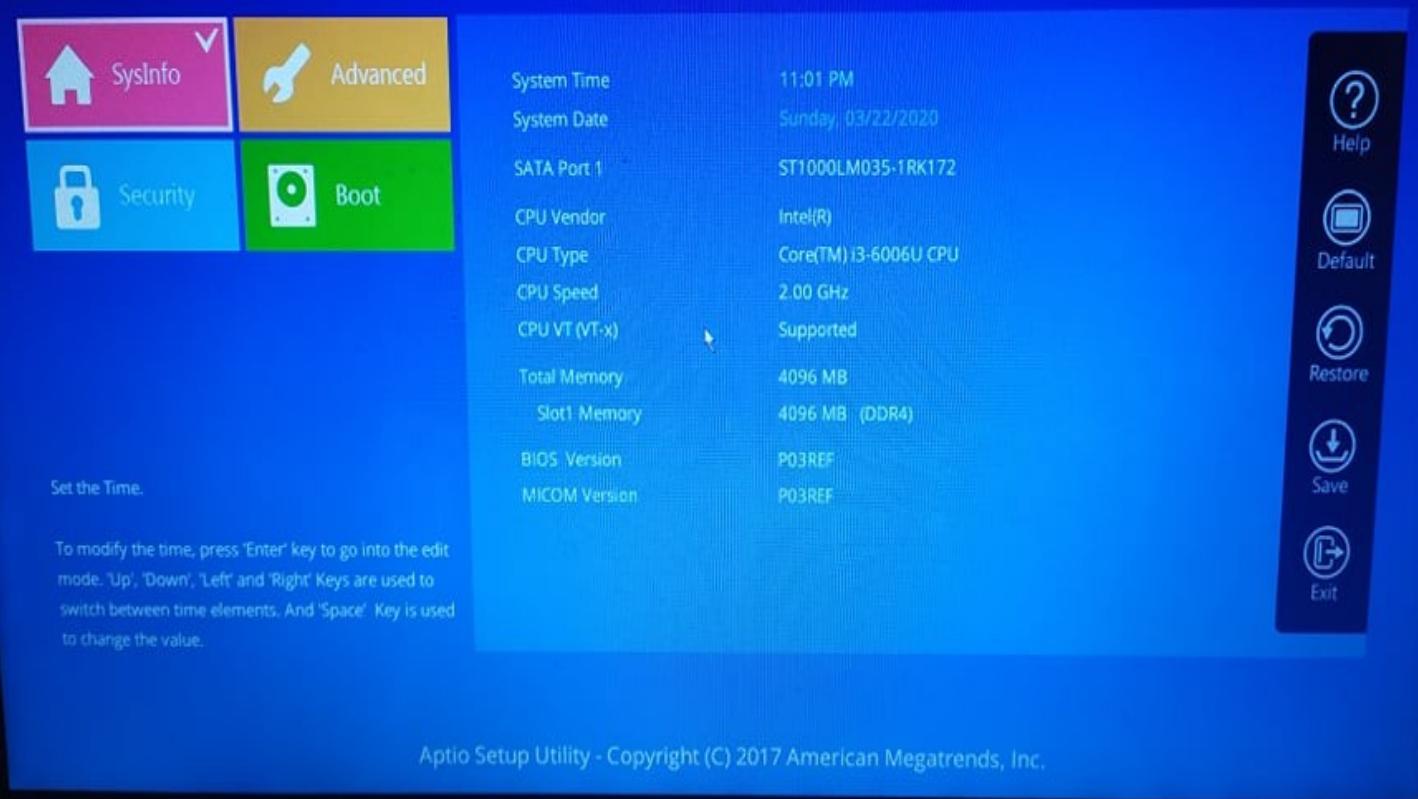
Processadores:

Marca: Intel → Informação presente no campo: CPU Vendor

Modelo: Core(TM) i3-6006U CPU → Informação presente no campo: CPU TYPE

Capacidade/CPU SPEED: 2.00GHz → Informação presente no campo: CPU MHZ

### Samsung BIOS Configuration 23:01 Sunday, March 22



As seguinte informação não aparece no SETUP do computador 1 para encontrar tais informações utilizei o comando lscpu no terminal:

Quantidade instalada: 500.045 MHz → Informação presente no campo: CPU MHZ

```
euller@euller:~$ lscpu
Arquitetura:          x86_64
Modo(s) operacional da CPU: 32-bit, 64-bit
Orden dos bytes:      Little Endian
CPU(s):              4
Lista de CPU(s) on-line: 0-3
Thread(s) per núcleo: 2
Núcleo(s) por soquete: 2
Soquete(s):          1
Nó(s) de NUMA:       1
ID de fornecedor:    GenuineIntel
Família da CPU:     6
Modelo:              78
Nome do modelo:      Intel(R) Core(TM) i3-6006U CPU @ 2.00GHz
Step:                3
CPU MHz:             500.045
CPU MHz máx.:        2000,0000
CPU MHz mín.:        400,0000
BogoMIPS:            3984,00
Virtualização:       VT-x
cache de L1d:        32K
cache de L1i:        32K
cache de L2:          256K
cache de L3:          3072K
CPU(s) de nó 0 NUMA: 0-3
Opções:               fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rd
tscp lm constant_tsc art arch_perfmon pebs bts rep_good nopl xtopology nonstop_tsc cpuid aperf mperf tsc_known freq pni pclmulqdq dtes64 monitor ds_cpl vmx est tm2 ssse3
sdbs fma cx16 xtpr pdcm pcid sse4_1 sse4_2 x2apic movbe popcnt tsc deadline_timer aes xsave avx f16c rdrandlahf_lm abm 3dnowprefetch cpuid fault epb invpcid single pti
tpr_shadow vnmi flexpriority ept vpid ept_ad fsgsbase tsc_adjust bmi1 avx2 smep bmi2 erms invpcid mpx rdseed adx smap clflushopt intel_pt xsaveopt xgetbv1 xsaves
dtherm arat pln pts hwp hwp_notify hwp_act_window hwp_epp
```

## Memória principal (RAM):

As seguintes informações não aparecem no setup do computador 1, para encontrar tais informações foi preciso utilizar comandos específicos no terminal do linux.

Marca: Smart Modular → Informação presente no campo: Manufacturer

Comando utilizado: sudo dmidecode -t 17

```
Handle 0x0016, DMI type 17, 40 bytes
Memory Device
  Array Handle: 0x0013
  Error Information Handle: Not Provided
  Total Width: 64 bits
  Data Width: 64 bits
  Size: 4096 MB
  Form Factor: SODIMM
  Set: None
  Locator: ChannelB-DIMM0
  Bank Locator: BANK 2
  Type: DDR4
  Type Detail: Synchronous Unbuffered (Unregistered)
  Speed: 2133 MT/s
  Manufacturer: Smart Modular
  Serial Number: 01E6CC86
  Asset Tag: 9876543210
  Part Number: SF464128CKHIDFSEG
  Rank: 1
  Configured Clock Speed: 2133 MT/s
  Minimum Voltage: Unknown
  Maximum Voltage: Unknown
  Configured Voltage: 1.2 V
```

Modelo: DDR4 → Informação presente no campo: Type

Comando utilizado: sudo dmidecode -t 17

```
Handle 0x0016, DMI type 17, 40 bytes
Memory Device
  Array Handle: 0x0013
  Error Information Handle: Not Provided
  Total Width: 64 bits
  Data Width: 64 bits
  Size: 4096 MB
  Form Factor: SODIMM
  Set: None
  Locator: ChannelB-DIMM0
  Bank Locator: BANK 2
  Type: DDR4
  Type Detail: Synchronous Unbuffered (Unregistered)
  Speed: 2133 MT/s
  Manufacturer: Smart Modular
  Serial Number: 01E6CC86
  Asset Tag: 9876543210
  Part Number: SF464128CKHIDFSEG
  Rank: 1
  Configured Clock Speed: 2133 MT/s
  Minimum Voltage: Unknown
  Maximum Voltage: Unknown
  Configured Voltage: 1.2 V
```

Capacidade: 3.858gb, ou seja, aproximadamente 4gb → Informação presente no campo: total

Comando utilizado: free -m

```
euller@euller:/$ free -m
              total        usada        livre  compart.  buff/cache  disponível
Mem.:          3858         2198         619          168          1040         1166
Swap:          2047           13        2034
```

Quantidade instalada: 2.198gb → Informação presente no campo: usada

```
euller@euller:~/
```

	total	usada	livre	compart.	buff/cache	disponível
Mem. :	3858	2198	619	168	1040	1166
Swap:	2047	13	2034			

Memória secundária(HD):

As seguintes informações não aparecem no setup do computador 1, para encontrar tais informações foi preciso utilizar comandos específicos no terminal do linux.

Marca: Seagate → Informação presente no campo: Fabricante

Comando utilizado: sudo lshw -C disk

```
euller@euller:~/
```

```
sudo lshw -C disk
```

```
*-disk
```

```
    descrição: ATA Disk
```

```
    produto: ST1000LM035-1RK1
```

```
    fabricante: Seagate
```

```
    ID físico: 0.0.0
```

```
    informações do barramento: scsi@0:0.0.0
```

```
    nome lógico: /dev/sda
```

```
    versão: SDM2
```

```
    serial: WDESAXMC
```

```
    tamanho: 931GiB (1TB)
```

```
    capacidades: gpt-1.00 partitioned partitioned:gpt
```

```
    configuração: ansiversion=5 guid=193fe9d6-2501-4a52-88ba-8a58a69d33fb logicalsectorsize=512 sectorsize=4096
```

Modelo: ATA disk → Informação presente no campo: Descrição

Comando utilizado: sudo lshw -C disk

```
euller@euller:~/
```

```
sudo lshw -C disk
```

```
*-disk
```

```
    descrição: ATA Disk
```

```
    produto: ST1000LM035-1RK1
```

```
    fabricante: Seagate
```

```
    ID físico: 0.0.0
```

```
    informações do barramento: scsi@0:0.0.0
```

```
    nome lógico: /dev/sda
```

```
    versão: SDM2
```

```
    serial: WDESAXMC
```

```
    tamanho: 931GiB (1TB) (não é possível entrar no menu de configuração do disco)
```

```
    capacidades: gpt-1.00 partitioned partitioned:gpt
```

```
    configuração: ansiversion=5 guid=193fe9d6-2501-4a52-88ba-8a58a69d33fb logicalsectorsize=512 sectorsize=4096
```

Capacidade: 931gb (~1tb) → Informação presente no campo: tamanho

Comando utilizado: sudo lshw -C disk

```
euller@euller:~/
```

```
sudo lshw -C disk
```

```
*-disk
```

```
    descrição: ATA Disk
```

```
    produto: ST1000LM035-1RK1
```

```
    fabricante: Seagate
```

```
    ID físico: 0.0.0
```

```
    informações do barramento: scsi@0:0.0.0
```

```
    nome lógico: /dev/sda
```

```
    versão: SDM2
```

```
    serial: WDESAXMC
```

```
    tamanho: 931GiB (1TB)
```

```
    capacidades: gpt-1.00 partitioned partitioned:gpt
```

```
    configuração: ansiversion=5 guid=193fe9d6-2501-4a52-88ba-8a58a69d33fb logicalsectorsize=512 sectorsize=4096
```

Quantidade instalada: 33.49774gb → Informação presente no campo: Usado

Comando utilizado: df -h

Para descobrir a quantidade utilizada do HD é preciso somar os valores presentes no campo “Usado”:

1.4mb + 33gb + 130mb + 0.004mb + 175mb + 92M + 92mb + 7.3mb + 0.036mb =

```
euller@euller:~$ df -h
Sist. Arq.      Tam. Usado Disp. Uso% Montado em
udev            1,9G    0   1,9G  0% /dev
tmpfs           386M  1,4M  385M  1% /run
/dev/sda2        916G  33G  837G  4% /
tmpfs           1,9G  132M  1,8G  7% /dev/shm
tmpfs           5,0M  4,0K  5,0M  1% /run/lock
tmpfs           1,9G    0   1,9G  0% /sys/fs/cgroup
/dev/loop0       175M  175M    0 100% /snap/skype/118
/dev/loop2       92M   92M    0 100% /snap/core/8592
/dev/loop1       92M   92M    0 100% /snap/core/8689
/dev/sdal        511M  7,3M  504M  2% /boot/efi
tmpfs           386M  36K  386M  1% /run/user/1000
                                                               33.49774gb
```

## Questão 5

Verifique se o sistema operacional do seu computador é multiusuário. Em caso positivo, crie dois novos usuários (user1 e user2). Acesse esse sistema, simultaneamente, com esses dois usuários. Para isso, utilize o login pelo console e o login remoto

O linux e o windows são multiusuários, ou seja, permitem que vários usuários utilizem o computador ao mesmo tempo.

Login pelo console:

Computador utilizado: Computador 1

O comando “su - user” é utilizado para acessar outro usuário pelo terminal do linux.

Então ao digitar “su - user1” a senha é requisitada, se a senha estiver correta o login é realizado com sucesso. O indicativo de que o login ocorreu está presente no terminal do linux, pois antes do login o terminal exibia “euller@euller”, após o login o terminal exibe “user1@euller”. Ao logar no user1, somente os arquivos disponíveis neste usuário poderão ser acessados, arquivos de outros usuários se tornam restritos.

```
euller@euller:~$ ls
Android 'Área de Trabalho' Documentos Downloads Imagens MEGASync Modelos Música public_html PÚblico snap Vídeos
euller@euller:~$ su - user1
Senha:
user1@euller:~$ ls
oi.txt
user1@euller:~$ 

euller@euller:~$ ls
Android 'Área de Trabalho' Documentos Downloads Imagens MEGASync Modelos Música public_html PÚblico snap Vídeos
euller@euller:~$ su - user2
Senha:
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

user2@euller:~$ ls
oi2.txt
user2@euller:~$
```

Login remoto:

Conexão linux → linux:

Computador utilizado: Computador 1

O comando “ssh user@hostname” é utilizado para acessar um determinado usuário de um computador específico, ou seja, tal comando permite o login remoto.

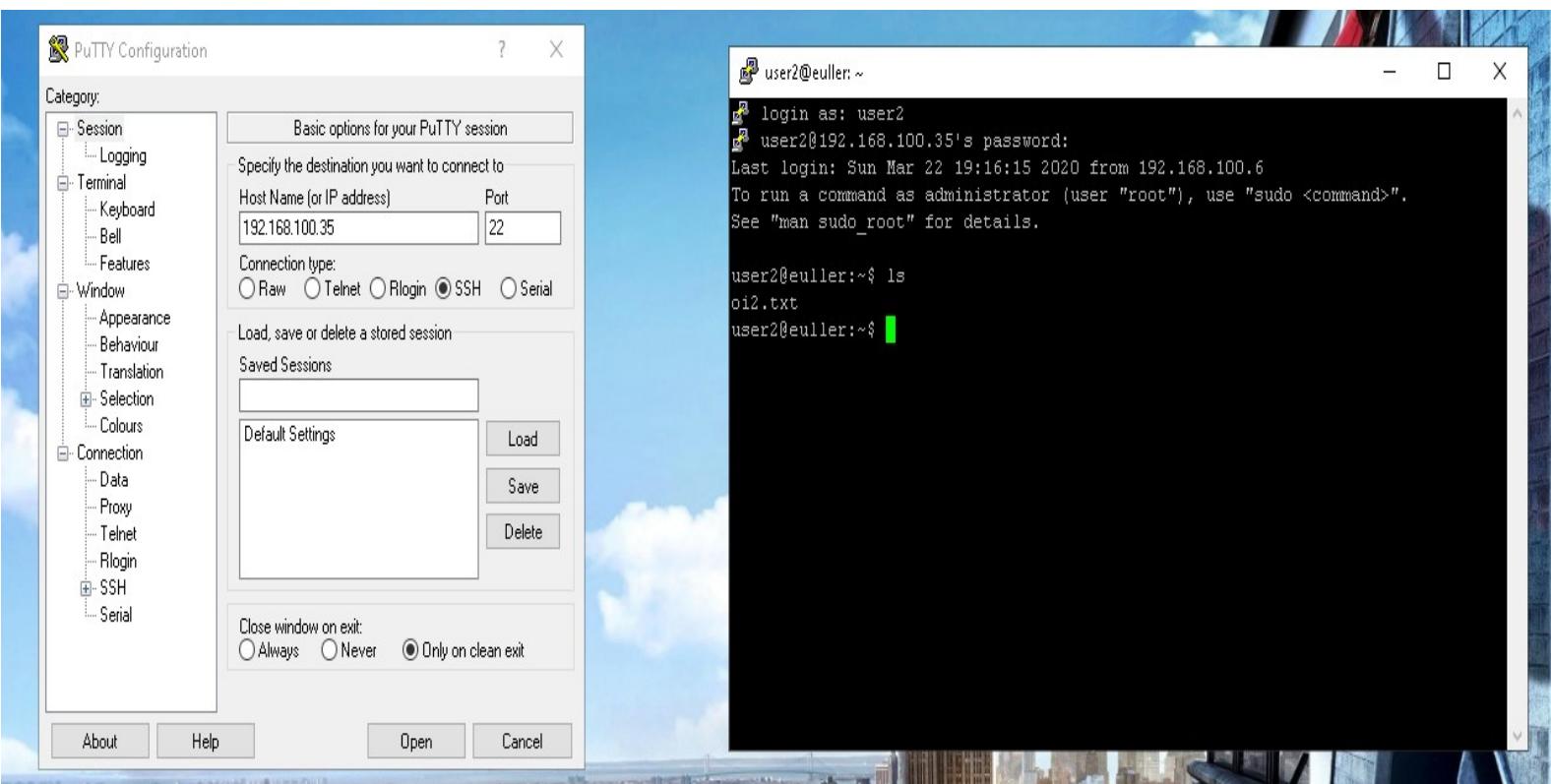
Então ao digitar “ssh [user2@192.168.100.35](mailto:user2@192.168.100.35)” em outro computador linux, a senha é requisitada, se a senha estiver correta o login é realizado com sucesso. O indicativo de que o login ocorreu está presente no terminal do linux, pois antes do login o terminal exibia “euller@euller”, após o login o terminal exibe “user2@euller”.

Conexão windows → linux

Computador utilizado: Computador 2

Para um computador windows realizar um login remoto em um computador linux é preciso utilizar o programa PuTTY.

Ao iniciar o programa uma interface gráfica será exibida com vários campos, contudo a única informação necessária no contexto atual é o endereço ip do computador linux. Após o endereço ip (192.168.100.35) ser fornecido e o botão “Open” ser apertado, um terminal shell é aberto, nesse terminal duas informações são requisitadas: o usuário (user2) e a senha. Se a senha estiver correta o login é realizado com sucesso. O indicativo de que o login ocorreu está presente no terminal, pois após o login ser realizado o terminal exibe “user2@euller”.



# Questão 6

Utilizando o programa shell, em linha de comando, execute diferentes programas no seu computador (ex. notepad.exe, gedit, etc.). Faça isso tanto no Linux quanto no Windows.

No terminal do linux:

Computador utilizado: Computador 1

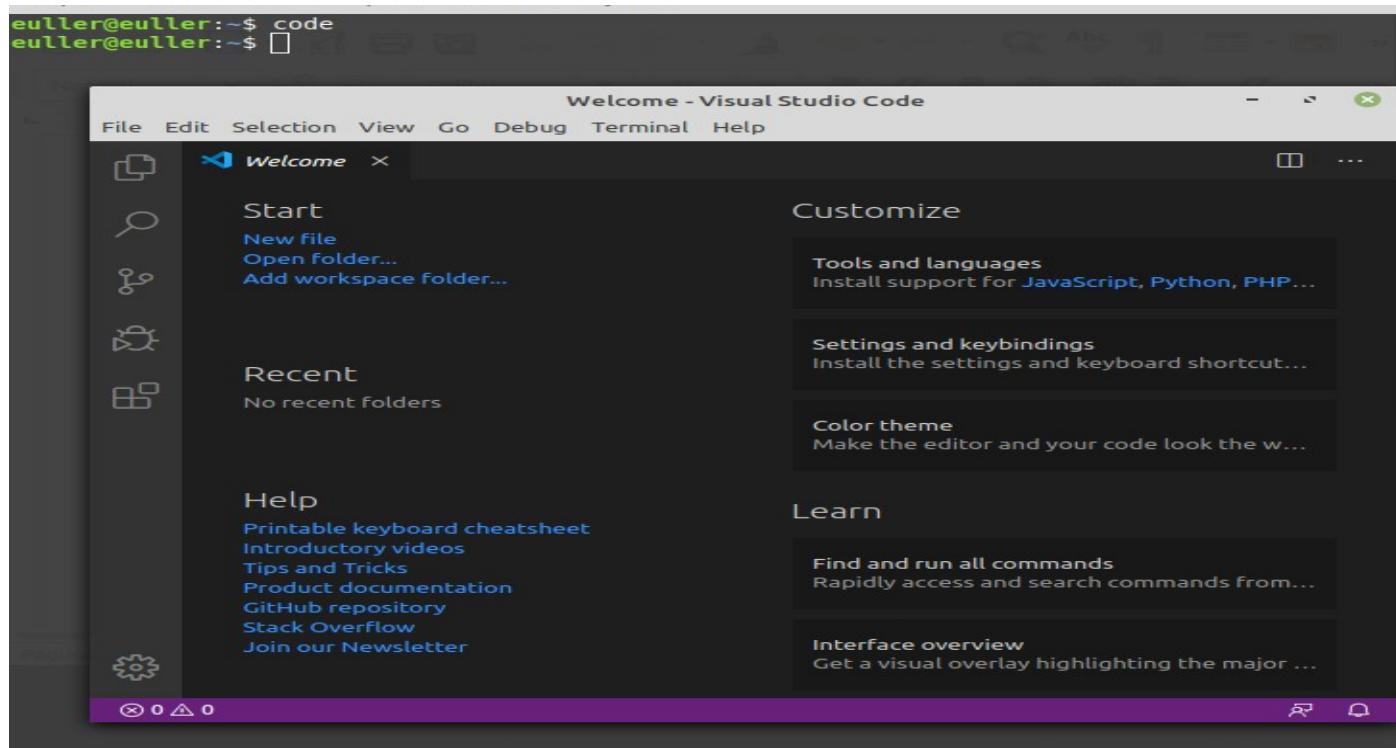
- 1) gedit oi.txt (abre o editor de texto gedit, abre o arquivo caso exista, cria o arquivo caso não exista)



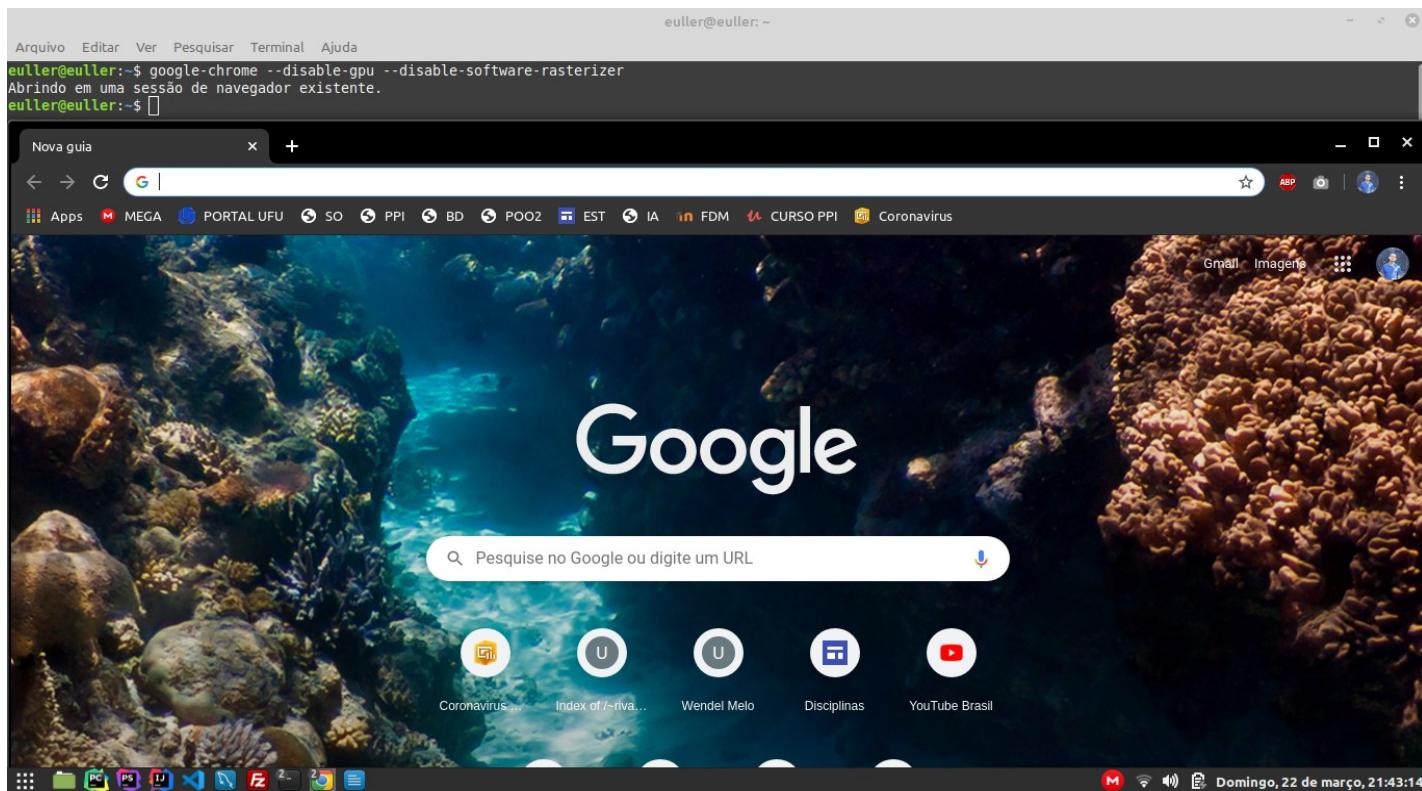
- 2) nano ola.c (abre o editor de texto nano no terminal, abre o arquivo caso exista, cria o arquivo caso não exista)



3) code (abre o Visual Studio Code)



4) google-chrome (abre o google-chrome)



5) python3 (abre o interpretador da linguagem python no terminal)

```
euller@euller:~$ python3
Python 3.6.9 (default, Nov  7 2019, 10:44:02)
[GCC 8.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

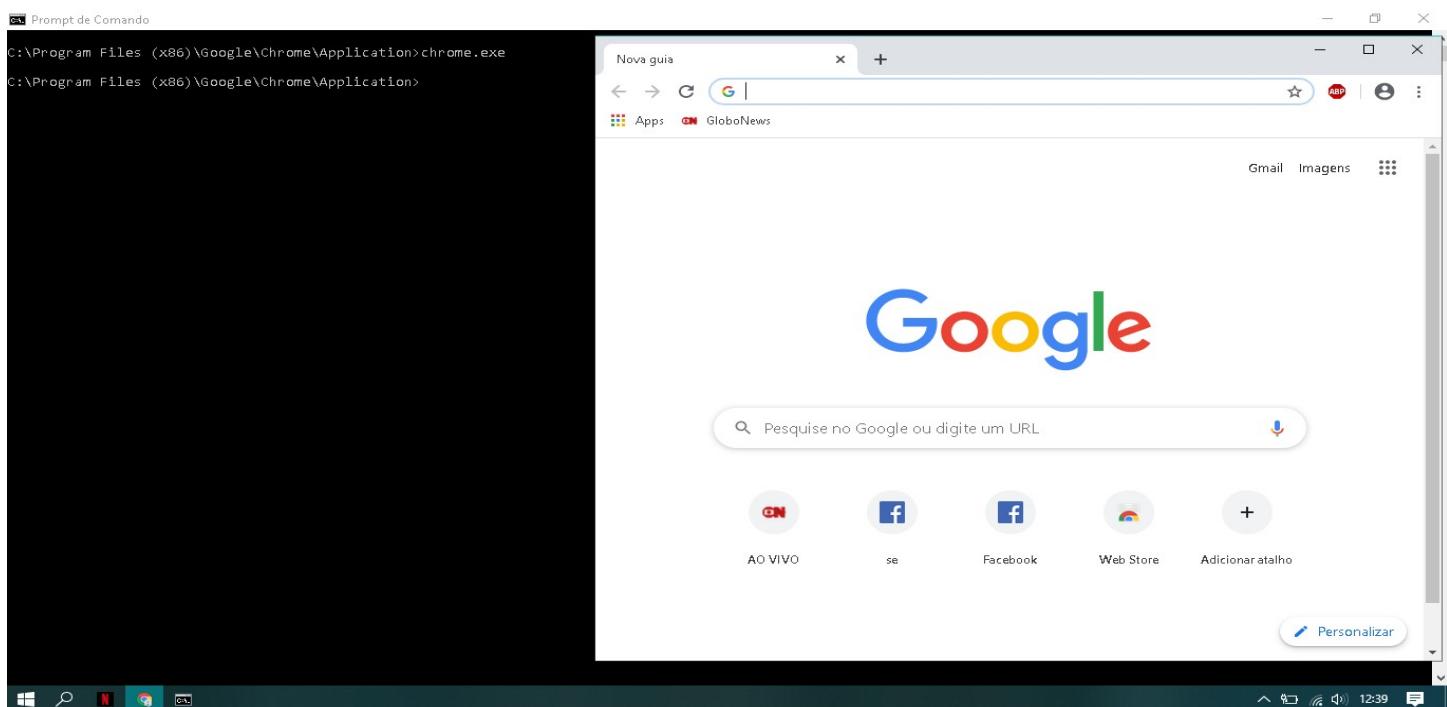
7. Utilizando o programa shell, em linha de comando, execute diferentes comandos.

No terminal do windows:

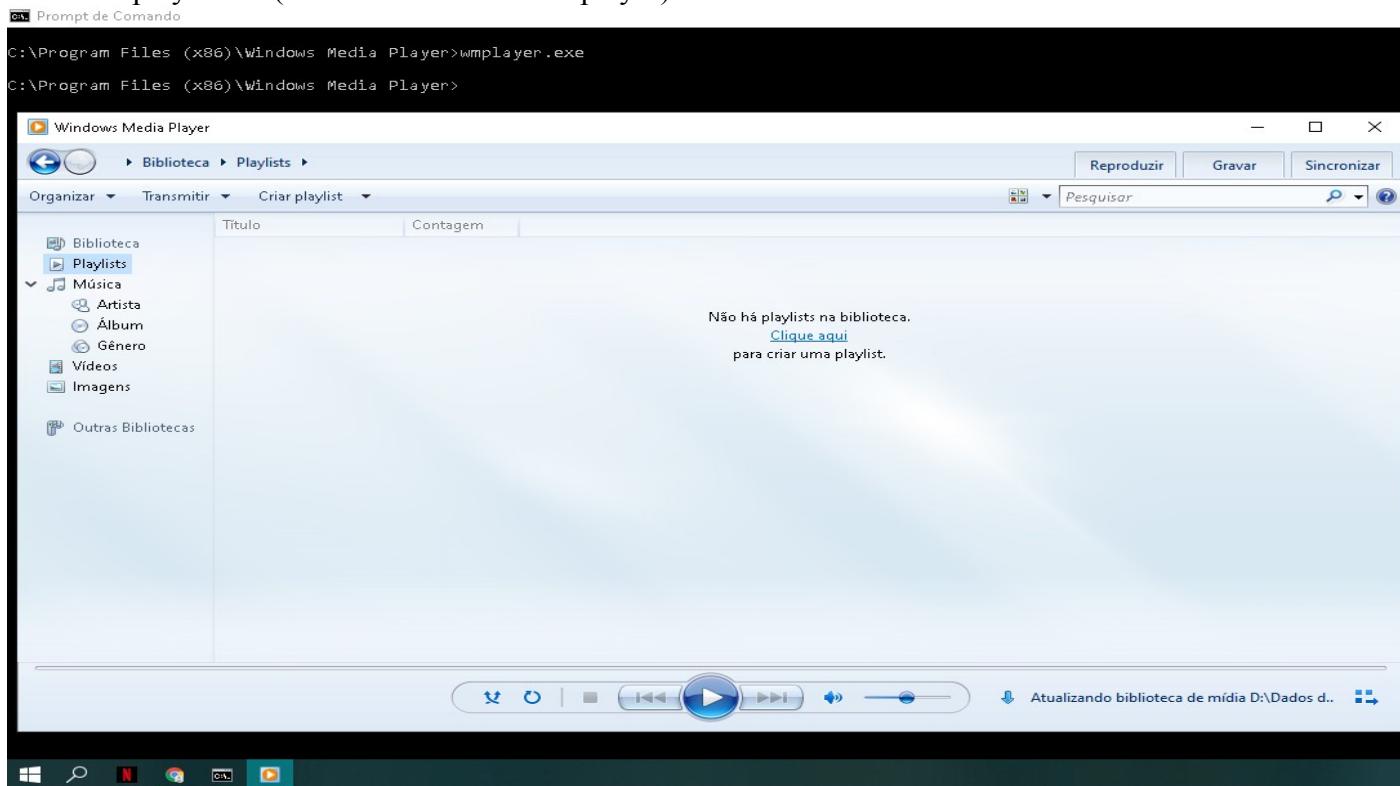
Computador utilizado: Computador 2

1) cd Program Files (x86)/ Google/Chrome/Application

chrome.exe (abre o google chrome)

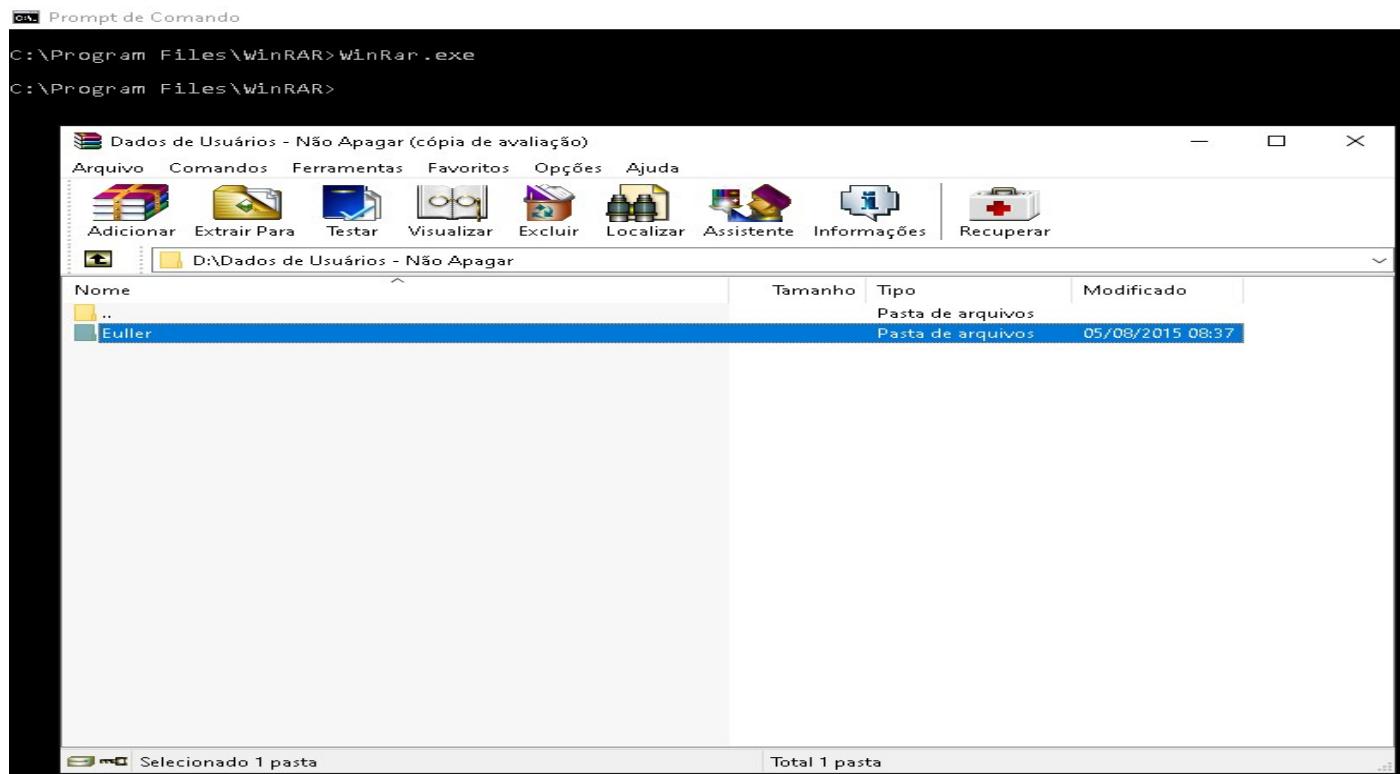


2) cd Program Files (x86)/Windows Media Player  
wmplayer.exe (abre o windows media player)



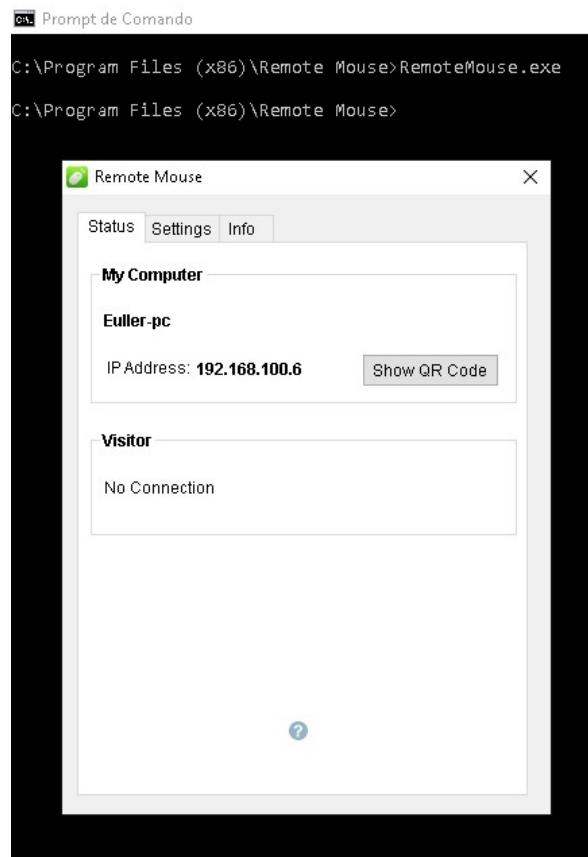
3) cd Program Files/WinRAR

WinRar.exe (abre o winrar)



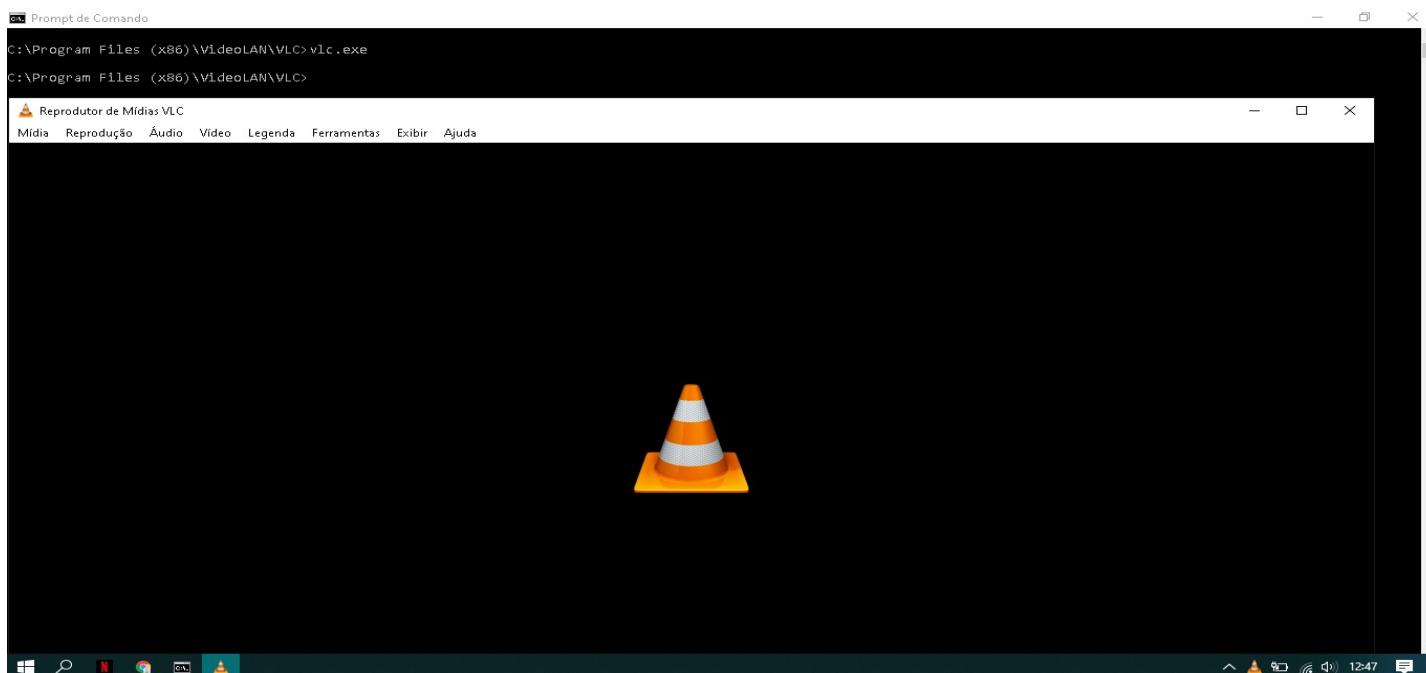
4) cd Program Files (x86)/Remote Mouse

RemoteMouse.exe (abre o remoteMouse)



5) cd Program Files (x86)/VideoLAN/VLC

vlc.exe (abre o vlc.exe)



# Questão 7

Utilizando o programa shell, em linha de comando, liste todos os processos existentes no computador. Faça isso tanto no Linux quanto no Windows.

No linux:

Computador utilizado: Computador 1

O comando top é utilizado para listar todos os processos existentes atualmente no computador.

Arquivo	Editar	Ver	Pesquisar	Terminal	Ajuda						
top - 17:11:56 up 4:10, 1 user, load average: 1,19, 0,98, 0,74											
Tarefas: 256 total, 3 em exec., 188 dormindo, 16 parado, 1 zumbi											
%CPU(s): 5,7 us, 1,4 sis, 0,0 ni, 92,4 oc, 0,0 ag, 0,0 ih, 0,5 is 0,0 tr											
KB mem : 3950636 total, 570880 livre, 2667152 usados, 712604 buff/cache											
KB swap: 2097148 total, 1949180 livre, 147968 usados, 766104 mem dispon.											
PID	USUARIO	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TEMPO+	COMANDO
2048	euller	20	0	5130280	401312	55680	S	15,2	10,2	36:23.15	chrome
1565	euller	20	0	3390460	114156	44896	R	7,3	2,9	19:56.30	cinnamon
1739	euller	20	0	624260	96312	44360	S	3,3	2,4	18:03.16	chrome
6083	euller	20	0	1448148	78556	26772	S	2,0	2,0	3:15.75	cinnamon-settin
8629	euller	20	0	667740	32068	21276	S	1,3	0,8	0:08.96	gnome-terminal-
11557	euller	20	0	44512	4288	3520	R	1,3	0,1	0:00.97	top
1018	mysql	20	0	1358656	68420	3216	S	0,7	1,7	0:14.58	mysqld
1032	root	20	0	771696	81128	60500	R	0,7	2,1	8:05.07	Xorg
1429	euller	20	0	449496	19628	14332	S	0,7	0,5	0:03.14	csd-keyboard
11181	euller	20	0	4791828	103244	72008	S	0,7	2,6	0:01.05	chrome
11452	root	20	0	0	0	0	I	0,7	0,0	0:00.08	kworker/2:0-mm_
1	root	20	0	225628	7104	4824	S	0,0	0,2	0:02.99	systemd
2	root	20	0	0	0	0	S	0,0	0,0	0:00.00	kthreadd
3	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	rcu_gp
4	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	rcu_par_gp
6	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker/0:0H-ev
8	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	mm_percpu_wq
9	root	20	0	0	0	0	S	0,0	0,0	0:00.89	ksoftirqd/0
10	root	20	0	0	0	0	I	0,0	0,0	0:32.35	rcu_sched
11	root	rt	0	0	0	0	S	0,0	0,0	0:00.17	migration/0
12	root	-51	0	0	0	0	S	0,0	0,0	0:00.00	idle_inject/0
14	root	20	0	0	0	0	S	0,0	0,0	0:00.00	cpuhp/0
15	root	20	0	0	0	0	S	0,0	0,0	0:00.00	cpuhp/1
16	root	-51	0	0	0	0	S	0,0	0,0	0:00.00	idle_inject/1
17	root	rt	0	0	0	0	S	0,0	0,0	0:00.17	migration/1
18	root	20	0	0	0	0	S	0,0	0,0	0:00.76	ksoftirqd/1
20	root	0	-20	0	0	0	I	0,0	0,0	0:02.11	kworker/1:0H-ev
21	root	20	0	0	0	0	S	0,0	0,0	0:00.00	cpuhp/2
22	root	-51	0	0	0	0	S	0,0	0,0	0:00.00	idle_inject/2
23	root	rt	0	0	0	0	S	0,0	0,0	0:00.18	migration/2
24	root	20	0	0	0	0	S	0,0	0,0	0:00.50	ksoftirqd/2
26	root	0	-20	0	0	0	I	0,0	0,0	0:00.00	kworker/2:0H-kb
27	root	20	0	0	0	0	S	0,0	0,0	0:00.00	cpuhp/3
28	root	-51	0	0	0	0	S	0,0	0,0	0:00.00	idle_inject/3
29	root	rt	0	0	0	0	S	0,0	0,0	0:00.18	migration/3
30	root	20	0	0	0	0	S	0,0	0,0	0:03.18	ksoftirqd/3

No windows:

Computador utilizado: Computador 2

O comando tasklist é utilizado para listar todos os processos existentes atualmente no computador.

Prompt de Comando				
C:\>tasklist   sort				
====	=====	=====	=====	=====
AdminService.exe	1320	Services	0	2.440 K
ApplicationFrameHost.exe	6908	Console	1	21.868 K
audiogd.exe	2336	Services	0	13.120 K
browser_broker.exe	9760	Console	1	7.468 K
chrome.exe	748	Console	1	33.188 K
chrome.exe	1180	Console	1	397.824 K
chrome.exe	1364	Console	1	37.356 K
chrome.exe	2308	Console	1	19.864 K
chrome.exe	3352	Console	1	212.100 K
chrome.exe	4248	Console	1	15.124 K
chrome.exe	4668	Console	1	40.660 K
chrome.exe	8864	Console	1	39.028 K
chrome.exe	9460	Console	1	160.132 K
chrome.exe	11784	Console	1	16.596 K
chrome.exe	12120	Console	1	82.104 K
chrome.exe	12220	Console	1	8.564 K
chrome.exe	12644	Console	1	6.708 K
cmd.exe	9952	Console	1	13.292 K
conhost.exe	11936	Console	1	20.424 K
csrss.exe	644	Services	0	1.912 K
csrss.exe	728	Console	1	2.708 K
ctfmon.exe	5724	Console	1	7.708 K
dasHost.exe	1092	Services	0	11.360 K
dllhost.exe	7196	Console	1	8.212 K
dwm.exe	1076	Console	1	31.432 K
explorer.exe	5912	Console	1	90.312 K
fontdrvhost.exe	76	Services	0	760 K
fontdrvhost.exe	1016	Console	1	3.928 K
GameScannerService.exe	3176	Services	0	5.492 K
GoogleCrashHandler.exe	1356	Services	0	576 K
GoogleCrashHandler64.exe	4764	Services	0	156 K
GoogleUpdate.exe	6224	Services	0	11.204 K
GoogleUpdate.exe	7128	Services	0	14.872 K
igfxCUIService.exe	1764	Services	0	1.936 K
igfxEM.exe	5384	Console	1	4.808 K
igfxHK.exe	5524	Console	1	2.752 K
LockApp.exe	6084	Console	1	25.940 K
lsass.exe	872	Services	0	10.292 K

## Questão 8

Utilizando o programa shell, em linha de comando, liste todos os arquivos existentes no seu computador. Faça isso tanto no Linux quanto no Windows.

## No Linux:

Computador utilizado: Computador 1

O comando `find` (sem nenhum parâmetro), digitado no diretório raiz do computador, lista todos os arquivos existentes no diretório atual, ou seja, lista todos os arquivos existentes no computador.

```
euller@euller:~/
```

Arquivo Editar Ver Pesquisar Terminal Ajuda

```
euller@euller:~/ find
```

```
./home
./home/euller
./home/euller/.sql
./home/euller/.ICEauthority
./home/euller/.gtkrc-2.0
./home/euller/.gradle
./home/euller/.gradle/caches
./home/euller/.gradle/caches/2.8
./home/euller/.gradle/caches/2.8/scripts
./home/euller/.gradle/caches/2.8/scripts/sync java lib9 ai8ns8d9hos2dh8ds5zw0h
./home/euller/.gradle/caches/2.8/scripts/sync java lib9 ai8ns8d9hos2dh8ds5zw0h/init
./home/euller/.gradle/caches/2.8/scripts/sync java lib9 ai8ns8d9hos2dh8ds5zw0h/init/metadata
./home/euller/.gradle/caches/2.8/scripts/sync java lib9 ai8ns8d9hos2dh8ds5zw0h/init/metadata.bin
./home/euller/.gradle/caches/2.8/scripts/sync java lib9 ai8ns8d9hos2dh8ds5zw0h/init/cache.properties
./home/euller/.gradle/caches/2.8/scripts/sync java lib9 ai8ns8d9hos2dh8ds5zw0h/init/classes
./home/euller/.gradle/caches/2.8/scripts/sync java lib9 ai8ns8d9hos2dh8ds5zw0h/init/classes/sync java lib9 ai8ns8d9hos2dh8ds5zw0h$ run closure1.class
./home/euller/.gradle/caches/2.8/scripts/sync java lib9 ai8ns8d9hos2dh8ds5zw0h.class
./home/euller/.gradle/caches/2.8/scripts/sync java lib9 ai8ns8d9hos2dh8ds5zw0h/init/cache.properties.lock
./home/euller/.gradle/caches/2.8/scripts/sync java lib9 ai8ns8d9hos2dh8ds5zw0h/cp init
./home/euller/.gradle/caches/2.8/scripts/sync java lib9 ai8ns8d9hos2dh8ds5zw0h/cp init/metadata
./home/euller/.gradle/caches/2.8/scripts/sync java lib9 ai8ns8d9hos2dh8ds5zw0h/cp init/metadata.bin
./home/euller/.gradle/caches/2.8/scripts/sync java lib9 ai8ns8d9hos2dh8ds5zw0h/cp init/cache.properties
./home/euller/.gradle/caches/2.8/scripts/sync java lib9 ai8ns8d9hos2dh8ds5zw0h/cp init/classes
./home/euller/.gradle/caches/2.8/scripts/sync java lib9 ai8ns8d9hos2dh8ds5zw0h/cp init/classes/sync java lib9 ai8ns8d9hos2dh8ds5zw0h$ run closure1.class
./home/euller/.gradle/caches/2.8/scripts/sync java lib9 ai8ns8d9hos2dh8ds5zw0h$closure1$closure2.class
./home/euller/.gradle/caches/2.8/scripts/sync java lib9 ai8ns8d9hos2dh8ds5zw0h/cp init/classes/sync java lib9 ai8ns8d9hos2dh8ds5zw0h.class
./home/euller/.gradle/caches/2.8/scripts/sync java lib9 ai8ns8d9hos2dh8ds5zw0h/cp init/cache.properties.lock
./home/euller/.gradle/caches/2.8/scripts/sync java lib9 b6b0v87mcg9eccew0dw8vgm7v
./home/euller/.gradle/caches/2.8/scripts/sync java lib9 b6b0v87mcg9eccew0dw8vgm7v/init
./home/euller/.gradle/caches/2.8/scripts/sync java lib9 b6b0v87mcg9eccew0dw8vgm7v/init/metadata
./home/euller/.gradle/caches/2.8/scripts/sync java lib9 b6b0v87mcg9eccew0dw8vgm7v/init/metadata.bin
./home/euller/.gradle/caches/2.8/scripts/sync java lib9 b6b0v87mcg9eccew0dw8vgm7v/init/cache.properties
./home/euller/.gradle/caches/2.8/scripts/sync java lib9 b6b0v87mcg9eccew0dw8vgm7v/init/classes
./home/euller/.gradle/caches/2.8/scripts/sync java lib9 b6b0v87mcg9eccew0dw8vgm7v/init/classes/sync java lib9 b6b0v87mcg9eccew0dw8vgm7v.class
./home/euller/.gradle/caches/2.8/scripts/sync java lib9 b6b0v87mcg9eccew0dw8vgm7v/init/classes/sync java lib9 b6b0v87mcg9eccew0dw8vgm7v$ run_closure1.class
./home/euller/.gradle/caches/2.8/scripts/sync java lib9 b6b0v87mcg9eccew0dw8vgm7v/init/cache.properties.lock
./home/euller/.gradle/caches/2.8/scripts/sync java lib9 b6b0v87mcg9eccew0dw8vgm7v/cp init
./home/euller/.gradle/caches/2.8/scripts/sync java lib9 b6b0v87mcg9eccew0dw8vgm7v/cp init/metadata
```

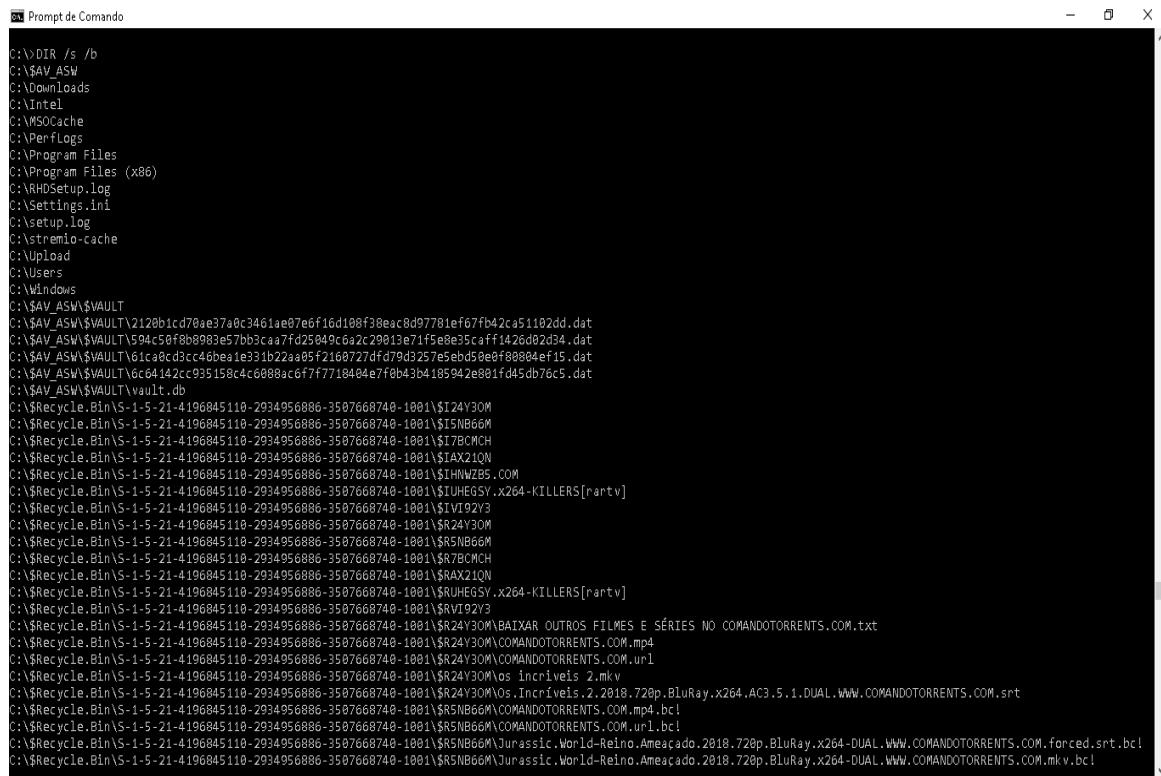
O comando tree digitado no diretório raiz do computador, lista em formato de árvore todos os arquivos existentes no diretório atual, ou seja, lista todos os arquivos existentes no computador.

```
euller@euller:~/tree
.
+- bin
|   +- archdetect
|   +- bash
|   +- brltty
|   +- btrfs
|   |   +- btrfsck -> btrfs
|   |   +- btrfs-debug-tree
|   |   +- btrfs-find-root
|   |   +- btrfs-image
|   |   +- btrfs-map-logical
|   |   +- btrfs-select-super
|   |   +- btrfsstune
|   |   +- btrfs-zero-log
|   +- bunzip2
|   +- busybox
|   +- bzcat
|   +- bzcmp -> bzdiff
|   +- bzdiff
|   +- bzgrep -> bzgrep
|   +- bzexe
|   +- bzfgrep -> bzgrep
|   +- bzgrep
|   +- bzip2
|   +- bzip2recover
|   +- bzless -> bzmore
|   +- bzmore
|   +- cat
|   +- chacl
|   +- chgrp
|   +- chmod
|   +- chown
|   +- chvt
|   +- cp
|   +- cpio
|   +- dash
|   +- date
|   +- dd
|   +- df
```

No Windows:

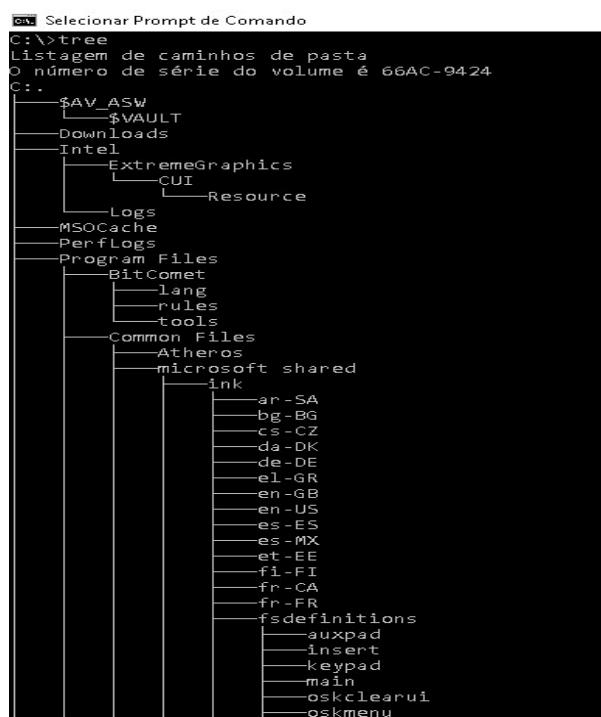
Computador utilizado: Computador 2

O comando DIR /s /b , digitado no diretório raiz do computador, lista todos os arquivos existentes no diretório atual, ou seja, lista todos os arquivos existentes no computador.



```
C:\>DIR /s /b
C:\$AV_ASW
C:\Downloads
C:Intel
C:\MSOCache
C:\PerLogs
C:\Program Files
C:\Program Files (x86)
C:\RHDSetup.log
C:\Settings.ini
C:\setup.log
C:\stremio-cache
C:\Upload
C:\Users
C:\Windows
C:\$AV_ASW\$VAULT
C:\$AV_ASW\$VAULT2120b1cd720ae37a0c3461ae07e6f16d108f38ec8d97781ef67fb42ca511b2dd.dat
C:\$AV_ASW\$VAULT594c50fb0b993e57bb3caa7fd25849c62c29013e71fe5e935caff1426002d34.dat
C:\$AV_ASW\$VAULT61ca0cdcc46bea1e31b22aa5f216b727df79d257e5ebd50e0f88804ef15.dat
C:\$AV_ASW\$VAULT6c64142cc935158c4c6888ac6f7f7718404e7f0b43b4185942e8001fd45db76c5.dat
C:\$AV_ASW\$VAULT7vault.db
C:\$Recycle.Bin\$-1-5-21-4196845110-2934956886-3507668740-1001\$I24Y30M
C:\$Recycle.Bin\$-1-5-21-4196845110-2934956886-3507668740-1001\$R5NB66M
C:\$Recycle.Bin\$-1-5-21-4196845110-2934956886-3507668740-1001\$T78CMCH
C:\$Recycle.Bin\$-1-5-21-4196845110-2934956886-3507668740-1001\$TAX21QN
C:\$Recycle.Bin\$-1-5-21-4196845110-2934956886-3507668740-1001\$IHNWZB5.COM
C:\$Recycle.Bin\$-1-5-21-4196845110-2934956886-3507668740-1001\$UHEGSV.x264-KILLERS[rarv]
C:\$Recycle.Bin\$-1-5-21-4196845110-2934956886-3507668740-1001\$V192Y3
C:\$Recycle.Bin\$-1-5-21-4196845110-2934956886-3507668740-1001\$R24Y30M
C:\$Recycle.Bin\$-1-5-21-4196845110-2934956886-3507668740-1001\$R5NB66M
C:\$Recycle.Bin\$-1-5-21-4196845110-2934956886-3507668740-1001\$R78CMCH
C:\$Recycle.Bin\$-1-5-21-4196845110-2934956886-3507668740-1001\$RAX21QN
C:\$Recycle.Bin\$-1-5-21-4196845110-2934956886-3507668740-1001\$RUHEGSV.x264-KILLERS[rarv]
C:\$Recycle.Bin\$-1-5-21-4196845110-2934956886-3507668740-1001\$V192Y3
C:\$Recycle.Bin\$-1-5-21-4196845110-2934956886-3507668740-1001\$R24Y30M\BATXAR OUTROS FILMES E SÉRIES NO COMANDOTORRENTS.COM.txt
C:\$Recycle.Bin\$-1-5-21-4196845110-2934956886-3507668740-1001\$R24Y30M\COMANDOTORRENTS.COM.mp4
C:\$Recycle.Bin\$-1-5-21-4196845110-2934956886-3507668740-1001\$R24Y30M\COMANDOTORRENTS.COM.url
C:\$Recycle.Bin\$-1-5-21-4196845110-2934956886-3507668740-1001\$R24Y30M\incríveis.2.2018.720p.BluRay.x264.AC3.5.1.DUAL.WWW.COMANDOTORRENTS.COM.srt
C:\$Recycle.Bin\$-1-5-21-4196845110-2934956886-3507668740-1001\$R5NB66M\COMANDOTORRENTS.COM.mp4.bcl
C:\$Recycle.Bin\$-1-5-21-4196845110-2934956886-3507668740-1001\$R5NB66M\COMANDOTORRENTS.COM.url.bcl
C:\$Recycle.Bin\$-1-5-21-4196845110-2934956886-3507668740-1001\$R5NB66M\Jurassic.World-Reino.Ameaçado.2018.720p.BluRay.x264-DUAL.WWW.COMANDOTORRENTS.COM.forced.srt.bcl
C:\$Recycle.Bin\$-1-5-21-4196845110-2934956886-3507668740-1001\$R5NB66M\Jurassic.World-Reino.Ameaçado.2018.720p.BluRay.x264-DUAL.WWW.COMANDOTORRENTS.COM.mkv.bcl
```

O comando tree digitado no diretório raiz do computador, lista em formato de árvore todos os arquivos existentes no diretório atual, ou seja, lista todos os arquivos existentes no computador.



```
C:\>tree
Listagem de caminhos de pasta
O número de série do volume é 66AC-9424
C:.
    $AV_ASW
        $VAULT
    Downloads
    Intel
        ExtremeGraphics
            CUI
                Resource
            Logs
        MSOCache
        PerfLogs
    Program Files
        BitComet
            lang
            rules
            tools
        Common Files
            Atheros
                microsoft shared
                    ink
                        ar-SA
                        bg-BG
                        cs-CZ
                        da-DK
                        de-DE
                        el-GR
                        en-GB
                        en-US
                        es-ES
                        es-MX
                        et-EE
                        fi-FI
                        fr-CA
                        fr-FR
                        fsdefinitions
                        auxpad
                        insert
                        keypad
                        main
                        oskclearui
                        oskmenu
```

## Questão 9

Utilizando ferramentas do sistema operacional, verifique o espaço livre em memória principal e secundária do computador. Faça isso tanto no Linux quanto no Windows.

No linux:

Computador utilizado: Computador 1

Memória principal(RAM):

Espaço livre: 619mb → Informação presente no campo: livre

Comando utilizado: free -m

```
euller@euller:~/Documents$ free -m
              total        usada        livre    compart.  buff/cache  disponível
Mem.:      3858         2198         619          168        1040       1166
Swap:     2047          13        2034
```

Memória secundaria(HD):

Espaço livre: 843.880gb → Informação presente no campo: Disp

Comando utilizado: df -h

Para descobrir a quantidade utilizada do HD é preciso somar os valores presentes no campo “Disp”:

$$1.9\text{gb} + 385\text{mb} + 837\text{gb} + 1.8\text{gb} + 5\text{mb} + 1.9\text{gb} + 504\text{mb} + 386\text{mb} \\ = 843.880\text{gb}$$

```
euller@euller:~/Documents$ df -h
Sist. Arq.      Tam. Usado Disp. Uso% Montado em
udev            1,9G   0   1,9G  0% /dev
tmpfs           386M  1,4M  385M  1% /run
/dev/sda2        916G  33G  837G  4% /
tmpfs           1,9G 132M  1,8G  7% /dev/shm
tmpfs           5,0M  4,0K  5,0M  1% /run/lock
tmpfs           1,9G   0   1,9G  0% /sys/fs/cgroup
/dev/loop0       175M  175M   0 100% /snap/skype/118
/dev/loop2       92M   92M   0 100% /snap/core/8592
/dev/loop1       92M   92M   0 100% /snap/core/8689
/dev/sdal        511M  7,3M  504M  2% /boot/efi
tmpfs           386M  36K  386M  1% /run/user/1000
```

No windows:

Computador utilizado: Computador 2

Memória principal(RAM):

Espaço livre: 929mb → Informação presente no campo: Memória física disponível

Comando utilizado: systeminfo

```
c:\ Prompt de Comando
Tipo de sistema: x64-based PC
Processador(es): 1 processador(es) instalado(s).
Versão do BIOS: [01]: Intel64 Family 6 Model 58 Stepping 9 GenuineIntel ~2400 Mhz
Pasta do Windows: Phoenix Technologies Ltd. P03RCB, 24/10/2013
Pasta do sistema: C:\WINDOWS
Inicializar dispositivo: C:\WINDOWS\system32
Localidade do sistema: \Device\HarddiskVolume2
Localidade de entrada: pt-br;Português (Brasil)
Fuso horário: pt-br;Português (Brasil)
(UTC-03:00) Brasília
Memória física total: 3.972 MB
Memória física disponível: 929 MB
Memória Virtual: Tamanho Máximo: 7.346 MB
Memória Virtual: Disponível: 2.278 MB
Memória Virtual: Em Uso: 5.068 MB
Local(is) de arquivo de paginação: C:\pagefile.sys
Domínio: WORKGROUP
Servidor de Logon: \\EULLER-PC
Hotfix(es): 8 hotfix(es) instalado(s).
[01]: KB4534132
[02]: KB4516115
[03]: KB4521863
[04]: KB4524569
[05]: KB4528759
[06]: KB4537759
[07]: KB4538674
[08]: KB4532693
Placa(s) de Rede: 2 NIC(s) instalado(s).
[01]: Qualcomm Atheros AR956x Wireless Network Adapter
      Nome da conexão: Wi-Fi
      DHCP ativado: Sim
      Servidor DHCP: 192.168.100.1
      Endereço(es) IP
      [01]: 192.168.100.6
      [02]: fe80::2c7b:404e:f29b:2872
[02]: Realtek PCIe FE Family Controller
      Nome da conexão: Ethernet
      Status: Mídia desconectada
      Extensão de Modo de Monitor VM: Sim
      Virtualização Habilitada no Firmware: Sim
      Conversão de Endereços de Segundo Nível: Sim
      Prevenção de Execução de Dados Disponível: Sim
Requisitos do Hyper-V:
C:\>
```

Memória secundaria(HD):

Espaço livre: 328gb → Informação presente no campo: FreeSpace

Comando utilizado: wmic logicaldisk get caption, size, freespace

C: 190736973824 bytes = 178gb | D: 160454139904 bytes = 150gb

178gb + 150gb = 328gb

```
C:\>wmic logicaldisk get caption, size, freespace
Caption  FreeSpace      Size
C:        190736973824  246580027392
D:        160454139904  234053693440
```

## Questão 10

Escreva um programa shell, em linha de comando, para Linux, o qual deve suportar os seguintes comandos:

CLS: apagar a tela.

DIR: listar os arquivos e subdiretórios (pastas) do diretório corrente.

DATE: mostrar a data corrente do sistema no formato DD/MM/AAAA.

TIME: mostrar a hora corrente do sistema no formato HH/MM/SS.

### Código:

```
#include <stdio.h>
#include <string.h>
#include <time.h>
#include <dirent.h>

struct tm *obterDataHora(){

    //Este ponteiro será utilizado para armazenar o endereço de uma struct tm.
    //Tal struct é utilizada para armazenar os dados relacionados a data e hora.

    struct tm *data_hora_atual;

    /* A struct tm foi definida da seguinte maneira na biblioteca time.h

    struct tm {

        int tm_sec; // Segundos (0-60)
        int tm_min; // Minutos (0-59)
        int tm_hour; // Horas (0-23)
        int tm_mday; // Dia do mês (1-31)
        int tm_mon; // Mês do ano (0-11)
        int tm_year; // Ano ( a partir de 1900 )
        int tm_wday; // Dia número da semana (0,1,2,3,4,5,6)
        int tm_yday; // Dia do ano ( 0-365 )
        int tm_isdst; // Horário de verão (!0)

    }; */

}
```

```

// Variável do tipo time_t que irá armazenar o tempo em segundos entre a data local e primeiro
de janeiro de 1970

time_t segundos;

// A função time é utilizada para obter o tempo em segundos
// Tal função possui como argumento o endereço da variável segundos.
// O tempo em segundos será armazenado no conteúdo do endereço da variável segundos.

time(&segundos);

// A função localtime é utilizada para converter o tempo em segundos para valores inteiros
// que representam cada atributo do tempo (segundos, minutos, hora, dia, mes, ano.. etc)
// Esses valores serão armazenados nos respectivos campos da struct tm
// Tal função retorna o endereço da struct com os dados de data e hora
// O ponteiro data_hora_atual do tipo struct tm recebe o endereço da struct que contém os
dados de data e hora

// Dessa maneira, é possível acessar todos os dados dessa struct
//O ponteiro data_hora_atual recebe o endereço da struct tm (ponteiro para a struct tm) que
contém os dados de date e hora

data_hora_atual = localtime(&segundos);

return data_hora_atual;
}

```

```

void TIME( struct tm *data_hora_atual ){
    char hora[20];
    // A função strftime formata a data para hh:mm:ss
    // Tal função recebe a hora por meio da variavel data_hora_atual
    // do tipo struct tm, acessa a hora, os minutos e os segundos armazenados na struct por meio
    // dos formatadores %H (hora), %M (minutos), %S (segundos), formata a data para hh:mm:ss e
    // finalmente armazena a hora formatada na variável hora

    strftime(hora, sizeof(hora), "%H:%M:%S\n", data_hora_atual);

    //Imprime a hora formatada
    printf("\nHora atual: %s\n", hora);

}

void DATE(struct tm *data_hora_atual){
    char data[20];

    // A função strftime formata a data para dd/mm/aaaa
    // Tal função recebe a data por meio da variavel data_hora_atual
    // do tipo struct tm, acessa o dia, o mes e o ano armazenados na struct por meio
    // dos formatadores %d (dia), %m (mes), %y (ano), formata a data para dd/mm/aaaa e
    // finalmente armazena a data formatada na variável data

    strftime(data, sizeof(data), "%d/%m/%y\n", data_hora_atual);

    //Imprime a data formatada
    printf("\nData atual: %s\n", data);

}

```

```

void DIRENT(){
    //O ponteiro dir do tipo DIR será utilizado para armazenar o endereço do diretório desejado
    DIR *dir;

    //Este ponteiro será utilizado para armazenar o endereço de uma struct dirent.
    //Tal struct é utilizada para armazenar os dados de um arquivo.

    struct dirent *lsdir = NULL;

    /* A struct dirent foi definida da seguinte maneira na biblioteca dirent

    struct dirent {
        ino_t      d_ino;
        off_t      d_off;
        unsigned short d_reclen; // Tamanho do arquivo
        unsigned char d_type;   // Tipo do arquivo
        char       d_name[256]; // Nome do arquivo
    };
    */

    // A função opendir abre o diretório determinado
    // Tal função retorna o endereço do diretório

    dir = opendir(".");
}

//A função readdir é utilizada para ler arquivos do diretório determinado
//Tal função possui como parâmetro um ponteiro que receberá o endereço do diretório que será
lido.

//Tal função retorna o endereço da struct dirent do arquivo atual presente no diretório
determinado

//A cada iteração, o próximo arquivo é lido

```

```
printf("\n");
while((lsdir = readdir(dir)) != NULL){

    char nome[256];

    //O nome do diretório é acessado por meio do campo d_name da struct dirent
    strcpy(nome, lsdir->d_name);

    //.. = diretório atual
    //.. = diretório pai
    //arquivo = arquivo oculto

    if(strcmp(nome, ".") != 0 && strcmp(nome, "..") != 0){

        printf("<DIR>      %s\n", nome);

    }

}

printf("\n");

//A função closedir fecha o diretório determinado
closedir(dir);

}
```

```
void CLS(){

    printf("\e[1;1H\e[2J");

    // Código de escape ansi
    // Ao utilizar um código de escape o terminal utilizado o executa,
    // alterando a cor do texto, movendo o cursor do mouse
    // ou pulando linhas

    // \e é o caractere de escape
    // Um caracter de escape é utilizado para indicar ao terminal que ele deve interpretar
    // os caracteres subsequentes a sequencia de escape de uma forma diferente da forma padrão
    // imposta pela linguagem utilizada
    // O caractere de escape \e é utilizado especificadamente para fazer com que a posição do cursor
    // do mouse
    // possa ser alterada e a tela possa ser limpa. Sem o caractere de escape \e, o terminal não
    // conseguiria perceber que aquela sequencia de caracteres poderia gerar ações, logo, ao executar
    // o código,
    // os caracteres seriam somente impressos.

    // [1;1h move o cursor para a posição (1,1)
    // [2J limpa a tela (pula linhas)
    // \e[1;1H\e[2J move o cursor para a posição(1,1) e limpa a tela de acordo com a posição do
    // cursor do mouse,
    // ou seja, como o mouse está na posição (1,1), /e[2J pulará linha até chegar na posição (1,1).

}
```

```

void executar(char comando[]){
    if(strcmp(comando, "CLS") == 0){
        CLS();
    }else if(strcmp(comando, "DIR") == 0){
        DIRENT();
    }else if(strcmp(comando, "DATE") == 0 || strcmp(comando, "TIME") == 0){
        //Este ponteiro será utilizado para armazenar o endereço de uma struct tm.
        //Tal struct é utilizada para armazenar os dados relacionados a data e hora.

        struct tm *data_hora_atual = obterDataHora();

        if(strcmp(comando, "DATE") == 0){
            DATE(data_hora_atual);
        }else if(strcmp(comando, "TIME") == 0){
            TIME(data_hora_atual);
        }
    }else{
        printf("\n%s : não encontrado\n", comando);
    }
}

```

```
}
```

```
int main(void){  
  
    char comando[10];  
  
    printf("\nShell v0.1 (Última atualização: 07/03/21) \n\n");  
    printf("Autor: Euller Henrique Bandeira Oliveira \n");  
    printf("Matrícula: 11821BSI210\n");  
    printf("Período: 5º\n");  
    printf("Curso: Sistemas De Informação\n");  
    printf("Universidade: Universidade Federal De Uberlândia\n\n");  
  
    while(1){  
        printf(">>> ");  
        fgets(comando, 10, stdin);  
        comando[strcspn(comando, "\n")] = '\0';  
        executar(comando);  
    }  
  
    return 0;  
}
```

## **Descrição:**

### **Main:**

Para iniciar a criação do shell foi preciso criar a função principal, ou seja, a int main.

A variável comando foi declarada nessa função. essa variável possui 10 caracteres do tipo char, ou seja, ela é um vetor de 10 caracteres. Tal variável será utilizada para armazenar o comando digitado pelo usuário.

Logo após a declaração da variável, um pequeno texto com as informações do shell será imprimido na tela.

Em seguida, um laço while infinito será iniciado.

Primeira instrução do laço:

A string “>>>” será impressa, tal string é utilizada somente para indicar ao usuário que ele deve digitar algum comando.

Segunda instrução do laço:

A função fgets (oriunda da biblioteca string.h) será utilizada para ler o comando digitado pelo usuário, a função possui 3 parâmetros: a variável que receberá o valor lido, o tamanho do valor que será lido e em que lugar o valor será lido. Nesse caso, o primeiro argumento é a variável comando, o segundo argumento é o valor 10 (tamanho do vetor comando) e o terceiro argumento é stdin (o teclado).

Terceira instrução do laço:

Objetivo: Eliminar o “\n” da string lida

Explicação: A função strcspn será chamada dentro do vetor comando com o objetivo de localizar em que índice desse vetor o “\n” está . Após o índice ser retornado pela função, o vetor comando com o índice em questão receberá a string “\0”, tal caractere indica o final de uma string.

A função strcspn (oriunda biblioteca string.h) possui dois parâmetros, o vetor que será percorrido e qual caractere será buscado. Nesse caso, o primeiro argumento é o vetor comando e o segundo argumento é o caractere “\n”.

Quarta instrução do laço:

A função executar será chamada, tal função possui como parâmetro um vetor de caracteres que receberá o comando que será executado. Nesse caso, o argumento é o vetor comando.

## **Executar:**

Essa função é constituída por uma sequência de 4 ifs que testará por meio da função strcmp se a string digitada pelo usuário corresponde à uma das seguintes strings: “CLS”, ‘DIR”, “DATE” ou “TIME”.

A função strcmp (oriunda da biblioteca string.h) possui dois parâmetros, ambos são vetores de caracteres, ou seja, strings. Essa função irá retornar 0 caso as duas strings forem iguais.

### **Se a condição (comando == “CLS”) do primeiro if for verdadeira:**

A função CLS será chamada.

### **Se a condição (comando == “DIR”) do segundo if for verdadeira:**

A função será DIR chamada.

### **Se a condição (comando == “DATE” || comando == “TIME”) do terceiro if for verdadeira:**

Esse if se inicia com a declaração do ponteiro data\_hora\_atual do tipo struct tm (oriunda da biblioteca time.h). Esse ponteiro será utilizado para armazenar o endereço de uma struct tm. Tal struct é utilizada para armazenar os dados relacionados a data e hora.

Tal ponteiro receberá o retorno da função obterDaraHora, essa função possui como retorno um ponteiro para a struct tm que contém os dados de data e hora.

A struct tm foi definida da seguinte maneira na biblioteca time.h:

```
struct tm {  
    int tm_sec; // Segundos (0-60)  
    int tm_min; // Minutos (0-59)  
    int tm_hour; // Horas (0-23)  
    int tm_mday; // Dia do mês (1-31)  
    int tm_mon; // Mês do ano (0-11)  
    int tm_year; // Ano ( a partir de 1900 )  
    int tm_wday; // Dia numérico da semana (0,1,2,3,4,5,6)  
    int tm_yday; // Dia do ano ( 0-365)  
    int tm_isdst; // Horário de verão (!0)  
};
```

### **Se a condição (comando == “DATE” ) do primeiro if for verdadeira:**

A função DATE será chamada, tal função possui como parâmetro um ponteiro do tipo struct tm que receberá o endereço da struct tm que contém os dados e data e hora. Nesse caso, o argumento é o ponteiro data\_hora\_atual.

### **Se a condição (comando == “TIME” ) do primeiro if for verdadeira:**

A função TIME será chamada, tal função possui como parâmetro um ponteiro do tipo struct tm que receberá o endereço da struct tm que contém os dados e data e hora. Nesse caso, o argumento é o ponteiro data\_hora\_atual.

### **Se o usuário não digitar nenhum comando válido:**

O comando digitado será impresso juntamente com a string “ :comando inválido”.

### **ObterDataHora:**

Essa função se inicia com a declaração do ponteiro data\_hora\_atual do tipo struct tm.

Logo após, a variável segundos do tipo time\_t é declarada. Tal variável irá armazenar o tempo em segundos entre a data local e primeiro de janeiro de 1970. Para utilizar esse tipo foi preciso incluir a biblioteca time.h no código por meio do #include <time.h> .

Logo em seguida, a função time (oriunda da biblioteca time.h) será chamada. Tal função é utilizada para obter o tempo em segundos. Essa função possui como parâmetro um ponteiro do tipo time\_t que receberá o endereço da variável que armazena os segundos atuais ( em relação a 01/01/1970). Nesse caso, o argumento é o endereço da variável segundos. O tempo em segundos será armazenado no conteúdo do endereço da variável segundos.

Logo após, o ponteiro data\_hora\_atual recebe o resultado da função localtime (oriunda da biblioteca time.h). Essa função possui como parâmetro um ponteiro um ponteiro do tipo time\_t que receberá o endereço da variável que armazena os segundos atuais ( em relação a 01/01/1970). Nesse caso, o argumento é o endereço da variável segundos. Tal função é utilizada para converter o tempo em segundos para valores inteiros que representam cada atributo do tempo (segundos, minutos, hora, dia, mês, ano... etc).

Logo em seguida, a função obterDataHora retorna o ponteiro data\_hora\_atual.

## CLS:

A seguinte string será imprimida na tela “\e[1;1H\e[2J”.

Essa string é um código de escape ansi. Quando se utiliza um código de espaço, o terminal o executa, alterando a cor do texto, movendo o cursor do mouse ou pulando linhas.

Explicação sobre a notação da string:

\e → É o caractere de escape

Um caractere de escape é utilizado para indicar ao terminal que ele deve interpretar os caracteres subsequentes a sequência de escape de uma forma diferente da forma padrão imposta pela linguagem utilizada.

O caractere de escape \e é utilizado especificadamente para fazer com que a posição do cursor do mouse possa ser alterada e a tela possa ser limpa. Sem o caractere de escape \e, o terminal não conseguiria perceber que aquela sequência de caracteres poderia gerar ações, logo, ao executar o código, os caracteres seriam somente impressos.

[1;1H → Move o cursor para a posição (1,1)

[2J → Limpa a tela (pula linhas)

e[1;1H\e[2J → Move o cursor para a posição (1,1) e limpa a tela de acordo com a posição do cursor do mouse, ou seja, como o mouse está na posição (1,1), \e[2J pulará linha até chegar na posição (1,1).

## DIRENT:

O ponteiro dir do tipo DIR será declarado. Esse ponteiro será utilizado para armazenar o endereço do diretório desejado. Para utilizar esse tipo foi preciso incluir a biblioteca dirent.h no código por meio do #include <dirent.h>.

O ponteiro lsdir do tipo struct dirent será declarado. Esse ponteiro será utilizado para armazenar o endereço de uma struct dirent. Tal struct é utilizada para armazenar os dados de um arquivo.

A struct dirent foi definida da seguinte maneira na biblioteca dirent:

```
struct dirent {  
    ino_t      d_ino;  
    off_t      d_off;  
    unsigned short d_reclen; // Tamanho do arquivo  
    unsigned char d_type;   // Tipo do arquivo  
    char       d_name[256]; // Nome do arquivo  
};
```

Em seguida, o ponteiro dir recebe o retorno da função opendir (oriunda da biblioteca dirent.h).

Essa função possui como parâmetro um vetor de caracteres que irá receber o caminho do diretório que será aberto. Nesse caso, se deseja abrir o diretório atual, ou seja, o diretório no qual o shell está sendo executado. Portanto, a função recebe “.” como argumento ( o “.” é utilizado para indicar o diretório atual). A função opendir retorna o endereço do diretório em questão, tal endereço é armazenado no ponteiro dir.

Logo após, um laço while com a seguinte condição será executado:

Se o ponteiro lsdir (esse ponteiro recebe o retorno da função readdir (oriunda da biblioteca dirent.h)) for diferente de null, o laço continua.

Essa função possui como parâmetro um ponteiro do tipo DIR que irá receber o endereço do diretório que será lido. Nesse caso, o argumento é o ponteiro dir. Tal função retorna o endereço da struct dirent do arquivo atual. A cada iteração o próximo arquivo é lido, quando não existir mais arquivo a função irá retornar null..

Ao iniciar o while duas variáveis serão declaradas: A variável nome com 256 caracteres do tipo char, ou seja, um vetor de 256 caracteres. Tal variável será utilizada para armazenar o nome do arquivo.

Em seguida, a função strcpy (oriunda da biblioteca string.h) será executada. Essa função possui dois parâmetros, o primeiro parâmetro é o vetor destino e o segundo parâmetro é o vetor origem. Nesse caso, o primeiro argumento é o vetor nome e o segundo argumento é o vetor d\_name da struct dirent presente no ponteiro lsdir, isto é, o segundo argumento é lsdir→d\_name. Essa função possui a utilidade de copiar a string presente no argumento 2 para o argumento 1.

Depois, um if é executado com as seguintes condições:

Se o nome do arquivo for diferente de “.” (diretório atual):

Por meio da função strcmp é testado se o vetor nome é diferente da string “.”, se for, a função retorna um valor diferente de 0.

&&

Se o nome do arquivo for diferente de “..” (diretório pai):

Por meio da função strcmp é testado se o vetor nome é diferente da string “..”, se for, a função retorna um valor diferente de 0.

Se o if atender todas as condições, o nome do arquivo será impresso.

Após o if e o while ser encerrado, um “\n” será imprimido na tela.

Por fim, a função closedir será executada. Essa função possui como parâmetro um ponteiro do tipo DIR que irá receber o endereço do diretório que será fechado. Nesse caso, esse argumento é o ponteiro dir.

## **DATE:**

A variável data com 20 caracteres do tipo char será declarada, ou seja, um vetor de 20 caracteres. Tal variável será utilizada para armazenar a data local.

Logo após, a função strftime (oriunda da biblioteca time.h) será executada. Tal função possui a utilidade de formatar datas e horários. Essa função possui quatro parâmetros, o primeiro parâmetro é o vetor que irá receber a string formatada, o segundo parâmetro é a quantidade de posições do vetor que serão utilizados para receber a string formatada, o terceiro parâmetro é a definição de como será a formatação e o quarto parâmetro é o ponteiro para a struct que contém os dados de data e hora. Nesse caso, o primeiro argumento é o vetor data, o segundo argumento é sizeof(data), o terceiro argumento é “%d/%m/%y\n” (%d (dia), %m (mês), %y (ano)) e o quarto argumento é o ponteiro data\_hora\_atual (ponteiro para a struct tm).

Posteriormente, a data será impressa na tela por meio da variável data.

## **TIME:**

A variável hora com 20 caracteres do tipo char será declarada, ou seja, um vetor de 20 caracteres. Tal variável será utilizada para armazenar a hora local.

Logo após, a função strftime (oriunda da biblioteca time.h) será executada. Tal função possui a utilidade de formatar datas e horários. Essa função possui quatro parâmetros, o primeiro parâmetro é o vetor que irá receber a string formatada, o segundo parâmetro é a quantidade de posições do vetor que serão utilizados para receber a string formatada, o terceiro parâmetro é a definição de como será a formatação e o quarto parâmetro é o ponteiro para a struct que contém os dados de data e hora. Nesse caso, o primeiro argumento é o vetor hora, o segundo argumento é sizeof(hora), o terceiro argumento é “%H:%M:%S\n” (%H (hora), %M (minutos), %S(segundos)) e o quarto argumento é o ponteiro data\_hora\_atual (ponteiro para a struct tm).

Posteriormente, o horário local será impresso na tela por meio da variável hora.

# Compilação e execução

## Compilação do SHELL:

```
euller@euller:~/Documentos/Study/UFU/4_QUARTO_PERIODO/Sistemas Operacionais>Listas$ gcc shell.c -o shell.exe  
euller@euller:~/Documentos/Study/UFU/4_QUARTO_PERIODO/Sistemas Operacionais>Listas$ █
```

## Execução do SHELL

```
euller@euller:~/Documentos/Study/UFU/4_QUARTO_PERIODO/Sistemas Operacionais>Listas$ gcc shell.c -o shell.exe  
euller@euller:~/Documentos/Study/UFU/4_QUARTO_PERIODO/Sistemas Operacionais>Listas$ ./shell.exe  
  
Shell v0.1 (Última atualização: 07/03/21)  
  
Autor: Euller Henrique Bandeira Oliveira  
Matrícula: 11821BSI210  
Período: 5º  
Curso: Sistemas De Informação  
Universidade: Universidade Federal De Uberlândia  
  
>>> █
```

## Execução do comando CLS:

```
euller@euller:~/Documentos/Study/UFU/4_QUARTO_PERIODO/Sistemas Operacionais>Listas$ ./shell.exe  
  
Shell v0.1 (Última atualização: 07/03/21)  
  
Autor: Euller Henrique Bandeira Oliveira  
Matrícula: 11821BSI210  
Período: 5º  
Curso: Sistemas De Informação  
Universidade: Universidade Federal De Uberlândia  
  
>>> TIME  
  
Hora atual: 20:04:51  
  
>>> DATE  
  
Data atual: 07/03/21  
  
>>> DIR  
  
<DIR> Atividade Prática de Fixação (Unidade 1).pdf  
<DIR> Atividade Prática de Fixação (Unidade 1).odt  
<DIR> Lista prática.odt  
<DIR> shell.c  
<DIR> shell.exe  
<DIR> .~lock.Atividade Prática de Fixação (Unidade 1).odt#  
  
>>> CLS█
```

```
>>> █
```

### Execução do comando DIR:

```
euller@euller:~/Documentos/Study/UFU/4_QUARTO_PERIODO/Sistemas Operacionais>Listas$ ./shell.exe
Shell v0.1 (Última atualização: 07/03/21)
Autor: Euller Henrique Bandeira Oliveira
Matrícula: 11821BSI210
Período: 5º
Curso: Sistemas De Informação
Universidade: Universidade Federal De Uberlândia

>>> DIR
<DIR> Atividade Prática de Fixação (Unidade 1).pdf
<DIR> Atividade Prática de Fixação (Unidade 1).odt
<DIR> Lista prática.odt
<DIR> shell.c
<DIR> shell.exe
<DIR> .~lock.Atividade Prática de Fixação (Unidade 1).odt#
>>> █
```

### Execução do comando TIME:

```
euller@euller:~/Documentos/Study/UFU/4_QUARTO_PERIODO/Sistemas Operacionais>Listas$ ./shell.exe
Shell v0.1 (Última atualização: 07/03/21)
Autor: Euller Henrique Bandeira Oliveira
Matrícula: 11821BSI210
Período: 5º
Curso: Sistemas De Informação
Universidade: Universidade Federal De Uberlândia

>>> TIME
Hora atual: 20:04:51
```

### Execução do comando DATE:

```
euller@euller:~/Documentos/Study/UFU/4_QUARTO_PERIODO/Sistemas Operacionais>Listas$ ./shell.exe
Shell v0.1 (Última atualização: 07/03/21)
Autor: Euller Henrique Bandeira Oliveira
Matrícula: 11821BSI210
Período: 5º
Curso: Sistemas De Informação
Universidade: Universidade Federal De Uberlândia

>>> DATE
Data atual: 07/03/21
>>> █
```