

QUESTÃO 1) Explique a função dos Sistemas de Arquivos. Adicionalmente, descreva a diferença entre o sistema de arquivos do linux e do windows.

É a parte do sistema operacional que lida com os arquivos. Ele fica responsável pela organização dos arquivos no disco, estrutura, forma de acesso, uso e como deve ser nomeado os arquivos.

No Windows 95, é usado o FAT16, o Windows 98 introduziu o FAT32 e os sistemas Windows após isso passaram a usar o NTFS, dando suporte ao FAT. O exFAT é uma extensão do FAT32 otimizado para flash drivers e sistemas de arquivos grandes quando o sistema de arquivos NTFS não for uma solução viável.

No Linux, há mais sistemas de arquivos, os mais usados são EXT3 e EXT4, outros exemplos são ReiserFS, XFS e JSF.

O Windows utiliza as extensões como forma de saber a qual programa deve executá-lo, enquanto outros, como o Linux, não obriga a necessidade de uma extensão, podendo ser usada pela demanda do usuário.

Em relação a caracteres:

Os sistemas FAT não são case sensitive, sendo que o FAT-16 suporta nomes de até 8 caracteres mais extensão de 1 a 3 caracteres e o FAT-32 suporta nomes de até 256 bits.

O NTFS utiliza caracteres Unicode, diferentemente do FAT que usa ASCII, apresenta suporte a criptografia e segurança.

O ext4 apresenta compatibilidade com o ext3 e também apresenta novos recursos de confiabilidade e desempenho, como a utilização de estruturas mais rápidas na organização de diretórios e redução de fragmentação de arquivos.

QUESTÃO 2) Existem quatro tipos de problemas que podem ocorrer na execução de processos concorrentes: trancamento (lockout), impasse (deadlock), inanição (starvation) e indeterminismo. Explique cada um deles dando exemplos de situações onde podem ocorrer.

Trancamento (lockout): quando vários processos aguardam um evento (liberar um recurso ocupado), reduzindo o desempenho. Um exemplo seria a fila de impressão com muitos processos nela esperando;

Impasse(deadlock): quando um conjunto de processos aguardam uma ação que somente outro processo do conjunto pode causar. Um exemplo seria quando o processo de um conjunto aguarda que o outro libere recurso para que possa ser executado, porém, como todos estão esperando recursos, nenhum será executado e causará impasse;

Inanição(starvation): quando um processo nunca executará devido a política de compartilhamento de recursos do sistema operacional. Um exemplo seria uma política que escolhe oferecer recurso para processos com menos exigência desse recurso, caso um processo exija bastante recurso, é possível que ele não seja executado devido a política;

Indeterminismo: quando o não conhecimento da ordem de acesso a um recurso compartilhado gera resultados não esperado. Isso pode acontecer quando threads tentam

acessar um recurso compartilhado em que uma está alterando uma variável antes de ser lida pela outra gerando resultado não desejado.

OBSERVAÇÃO: A questão 3 e 4 foram executados no sistema operacional "Ubuntu 18.04.2 LTS" no kernel "4.15.0-51-generic" e o gcc 7.4.0

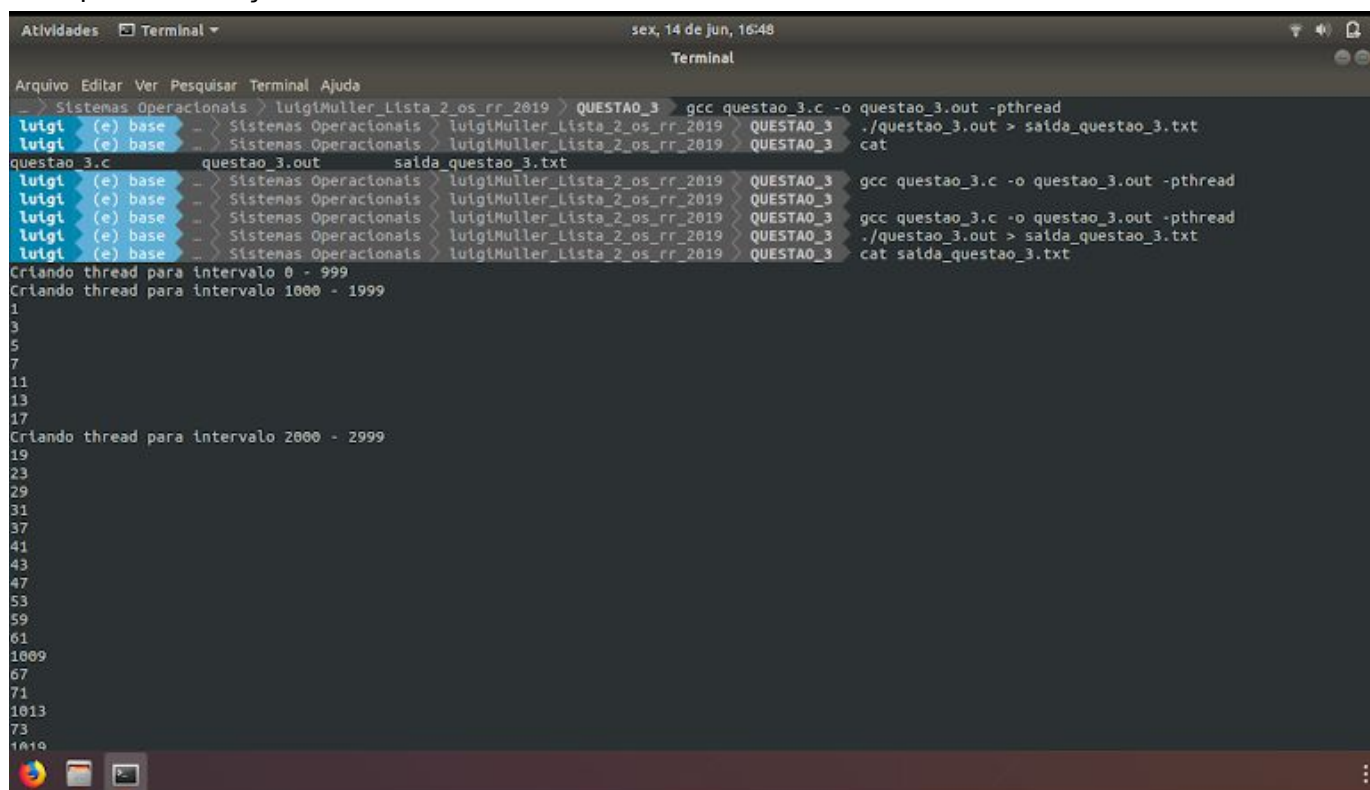
QUESTÃO 3) Faça um programa que imprima os números primos existentes entre 0 e 99999. UTILIZE THREADS. Dica: para cada faixa de mil valores crie uma thread e dispare o processo para cada uma delas.

A questão 3 foi compilada com o seguinte comando no terminal:
gcc questao_3.c -o questao_3.out -pthread

E executada com o seguinte comando:
./questao_3.out > saida_questao_3.txt

Esse comando executa o comando e joga a saída para o arquivo "saida_questao_3.txt"

Exemplo de execução:



```
Atividades Terminal sex, 14 de jun, 16:48
Terminal
Arquivo Editar Ver Pesquisar Terminal Ajuda
luigi (e) base ~ > Sistemas Operacionais > luigiMuller_Lista_2_os_rr_2019 > QUESTAO_3 gcc questao_3.c -o questao_3.out -pthread
luigi (e) base ~ > Sistemas Operacionais > luigiMuller_Lista_2_os_rr_2019 > QUESTAO_3 ./questao_3.out > saida_questao_3.txt
questao_3.c questao_3.out saida_questao_3.txt
luigi (e) base ~ > Sistemas Operacionais > luigiMuller_Lista_2_os_rr_2019 > QUESTAO_3 gcc questao_3.c -o questao_3.out -pthread
luigi (e) base ~ > Sistemas Operacionais > luigiMuller_Lista_2_os_rr_2019 > QUESTAO_3 cat
luigi (e) base ~ > Sistemas Operacionais > luigiMuller_Lista_2_os_rr_2019 > QUESTAO_3 gcc questao_3.c -o questao_3.out -pthread
luigi (e) base ~ > Sistemas Operacionais > luigiMuller_Lista_2_os_rr_2019 > QUESTAO_3 ./questao_3.out > saida_questao_3.txt
luigi (e) base ~ > Sistemas Operacionais > luigiMuller_Lista_2_os_rr_2019 > QUESTAO_3 cat saida_questao_3.txt
Criando thread para intervalo 0 - 999
Criando thread para intervalo 1000 - 1999
2
3
5
7
11
13
17
Criando thread para intervalo 2000 - 2999
19
23
29
31
37
41
43
47
53
59
61
1009
67
71
1013
73
1719
```

QUESTÃO 4) Implemente um programa que simule um lista de tarefas, usando listas encadeada por meio da biblioteca linux/list.h. Ver <https://github.com/torvalds/linux/blob/master/include/linux/list.h>

Os códigos são compilados com os seguinte comando(obs: o caminho da pasta e a pasta não podem ter espaços):

```
gcc TODO_USER.c -o test
make
```

O primeiro comando compila o TODO_USER.c que é o programa que será executado no espaço de usuário e o segundo compila o TODO_LKM.c que será o módulo executado no kernel.

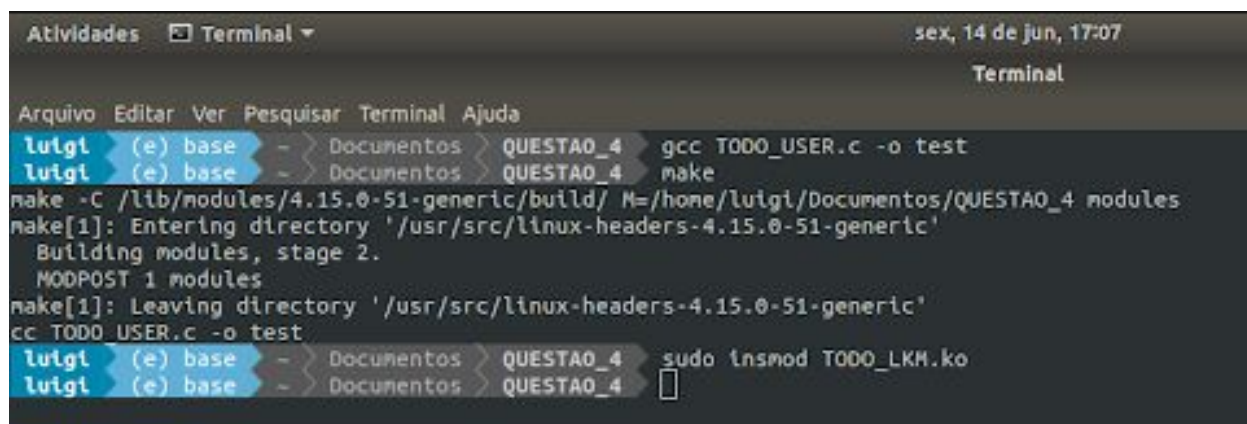
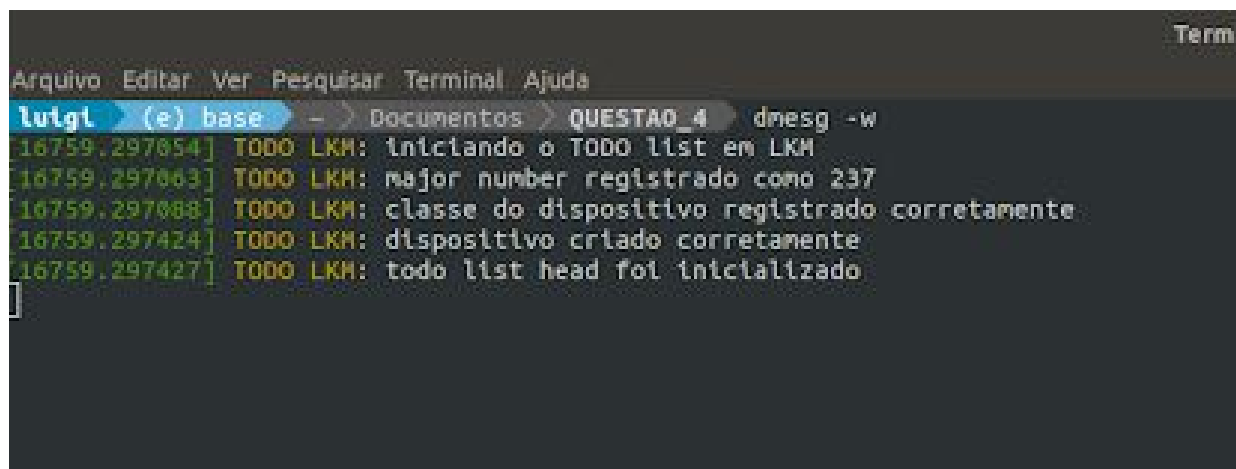
Agora os comandos para executar o programa são:

```
sudo insmod TODO_LKM.ko
sudo ./test
```

Pressione ENTER para ele inserir as tarefas dentro do módulo no kernel. E pressione ENTER para ele enviar as tarefas do módulo para o usuário. Ele exibirá as tarefas no terminal normalmente e encerrará o programa automaticamente.

```
sudo rmmod TODO_LKM
```

Exemplo de execução:

A terminal window titled 'Terminal' with a menu bar (Arquivo, Editar, Ver, Pesquisar, Terminal, Ajuda) and a status bar (sex, 14 de jun, 17:07). The user 'luigi' is in the directory '(e) base' with a path indicator showing 'Documentos > QUESTAO_4'. The terminal shows the execution of 'gcc TODO_USER.c -o test', 'make', and 'sudo insmod TODO_LKM.ko'. The output of 'make' shows it entering the directory '/usr/src/linux-headers-4.15.0-51-generic' and building modules. The output of 'insmod' shows 'cc TODO_USER.c -o test'.A terminal window titled 'Terminal' with a menu bar (Arquivo, Editar, Ver, Pesquisar, Terminal, Ajuda) and a status bar (Term). The user 'luigi' is in the directory '(e) base' with a path indicator showing 'Documentos > QUESTAO_4'. The terminal shows the execution of 'dmesg -w', which displays kernel messages from the 'TODO LKM' module. The messages include: 'TODO LKM: iniciando o TODO list em LKM', 'TODO LKM: major number registrado como 237', 'TODO LKM: classe do dispositivo registrado corretamente', 'TODO LKM: dispositivo criado corretamente', and 'TODO LKM: todo list head foi inicializado'.

Quando o comando insmod é inserido o log do Kernel notifica sobre a criação do módulo, nesse caso a módulo foi criado junto com a variável que faz referência a lista.

```
Atividades Terminal sex, 14 de jun, 17:07
Terminal
Arquivo Editar Ver Pesquisar Terminal Ajuda
lulgt (e) base - > Documentos > QUESTAO_4 gcc TODO_USER.c -o test
lulgt (e) base - > Documentos > QUESTAO_4 make
make -C /lib/modules/4.15.0-51-generic/build/ M=/home/lulgt/Documents/QUESTAO_4 modules
make[1]: Entering directory '/usr/src/linux-headers-4.15.0-51-generic'
Building modules, stage 2.
MODPOST 1 modules
make[1]: Leaving directory '/usr/src/linux-headers-4.15.0-51-generic'
cc TODO_USER.c -o test
lulgt (e) base - > Documentos > QUESTAO_4 sudo insmod TODO_LKM.ko
lulgt (e) base - > Documentos > QUESTAO_4 sudo ./test
Começando o código de teste do dispositivo...
Pressione ENTER para as cinco tarefas serem inseridas no LKM:

Escrevendo as tarefas no dispositivo.
Pressione ENTER para ler as tarefas a fazer...

Lendo do dispositivo...
Mensagem do dispositivo: [Abrir porta
Beber cafe
Comer lasanha
Dormir
Estudar
]
Fim do programa, TCHAU
```

```
Terminal
Arquivo Editar Ver Pesquisar Terminal Ajuda
lulgt (e) base - > Documentos > QUESTAO_4 dmesg -w
[16759.297854] TODO LKM: iniciando o TODO list em LKM
[16759.297863] TODO LKM: major number registrado como 237
[16759.297888] TODO LKM: classe do dispositivo registrado corretamente
[16759.297424] TODO LKM: dispositivo criado corretamente
[16759.297427] TODO LKM: todo list head foi inicializado
[16772.238892] TODO LKM: dispositivo foi aberto 1 vez(es)
[16779.116904] TODO LKM: enviado 52 caracteres para o usuario
[16779.117187] TODO LKM: dispositivo fechado corretamente
```

Agora o programa que fará a comunicação com o usuário e o módulo no kernel é executado. O log informa que o dispositivo foi aberto, enviou os caracteres com a lista de tarefas para o usuário e foi fechado, porém, o módulo ainda não saiu do kernel e não apagou as tarefas.

```
Atividades Terminal sex, 14 de jun, 17:07
Terminal
Arquivo Editar Ver Pesquisar Terminal Ajuda
luigi (e) base ~ > Documentos > QUESTAO_4 gcc TODO_USER.c -o test
luigi (e) base ~ > Documentos > QUESTAO_4 make
make -C /lib/modules/4.15.0-51-generic/build/ M=/home/luigi/Documentos/QUESTAO_4 modules
make[1]: Entering directory '/usr/src/linux-headers-4.15.0-51-generic'
Building modules, stage 2.
MODPOST 1 modules
make[1]: Leaving directory '/usr/src/linux-headers-4.15.0-51-generic'
cc TODO_USER.c -o test
luigi (e) base ~ > Documentos > QUESTAO_4 sudo insmod TODO_LKM.ko
luigi (e) base ~ > Documentos > QUESTAO_4 sudo ./test
Comecando o codigo de teste do dispositivo...
Pressione ENTER para as cinco tarefas serem inseridas no LKM:

Escrevendo as tarefas no dispositivo.
Pressione ENTER para ler as tarefas a fazer...

Lendo do dispositivo...
Mensagem do dispositivo: [Abrir porta
Beber cafe
Comer lasanha
Dormir
Estudar
]
Fim do programa, TCHAU
luigi (e) base ~ > Documentos > QUESTAO_4 sudo rmmod TODO_LKM
luigi (e) base ~ > Documentos > QUESTAO_4
```

```
Terminal
Arquivo Editar Ver Pesquisar Terminal Ajuda
luigi (e) base ~ > Documentos > QUESTAO_4 dmesg -w
[16759.297054] TODO LKM: iniciando o TODO list em LKM
[16759.297063] TODO LKM: major number registrado como 237
[16759.297088] TODO LKM: classe do dispositivo registrado corretamente
[16759.297424] TODO LKM: dispositivo criado corretamente
[16759.297427] TODO LKM: todo list head foi inicializado
[16772.238892] TODO LKM: dispositivo foi aberto 1 vez(es)
[16779.116904] TODO LKM: enviado 52 caracteres para o usuario
[16779.117187] TODO LKM: dispositivo fechado corretamente
[16786.602367] TODO LKM: apagando recado: Abrir porta
[16786.602372] TODO LKM: apagando recado: Beber cafe
[16786.602374] TODO LKM: apagando recado: Comer lasanha
[16786.602376] TODO LKM: apagando recado: Dormir
[16786.602378] TODO LKM: apagando recado: Estudar
[16786.602380] TODO LKM: se nao aparecer mensagem de recado apagado entao a lista esta vazia
[16786.602382] TODO LKM: saindo de boas do kernel!

```

O módulo é removido do kernel e o log informa que ele apagou as tarefas e está saindo do kernel.

Referências:

BOS, Herbert; TANENBAUM, Andrew S. **Sistemas Operacionais Modernos**. 4ª Edição. Pearson, 2016. 778 p.

ALENCAR, Felipe. **Entenda o que é sistema de arquivos e sua utilidade no PC e no celular**. TechTudo, 2016. Disponível em: <<https://www.techtudo.com.br/dicas-e-tutoriais/noticia/2016/02/entenda-o-que-e-sistema-de-arquivos-e-sua-utilidade-no-pc-e-no-celular.html>>. Acesso em: 07 de jun. 2019.

exFAT. Wikipédia, 2018. Disponível em: <<https://pt.wikipedia.org/wiki/ExFAT>>. Acesso em: 07 de jun. 2019.

File Allocation Table. Wikipédia, 2018. Disponível em: <https://pt.wikipedia.org/wiki/File_Allocation_Table>. Acesso em: 07 de jun. 2019.

NTFS. Wikipédia, 2018. Disponível em: <<https://pt.wikipedia.org/wiki/NTFS>>. Acesso em: 07 de jun. 2019.

JONES, M. **Anatomia do Ext4**. IBM Developer, 2009. Disponível em: <<https://www.ibm.com/developerworks/br/library/l-anatomy-ext4/index.html>>. Acesso em: 07 jun. 2019.

MOLLOY, Derek. **Writing a Linux Kernel Module — Part 2: A Character Device**. derekmolloy.ie, 2015. Disponível em: <<http://derekmolloy.ie/writing-a-linux-kernel-module-part-2-a-character-device/>>. Acesso em: 07 jun. 2019.

Linux Kernel Linked List Explained. 2005. Disponível em: <<http://isis.poly.edu/kulesh/stuff/src/klist/>>. Acesso em: 10 jun. 2019.

Creating linked list in liinux Kernel using the list. 2014. Disponível em: <<http://tuxthink.blogspot.com/2014/02/creating-linked-list-in-liinux-kernel.html>>. Acesso em: 11 jun. 2019.