

SISTEMAS COMPUTACIONAIS E SEGURANÇA

AGOSTO

15/08 – Início das Aulas
22/08 – Aula 1
29 a 31/08 – **TECHWEEK**

SETEMBRO

05/09 – Aula 2
12/09 – Aula 3
19/09 – Aula 4
26/09 – Aula 5

OUTUBRO

03/10 – Aula 6
10/10 – Aula 7
17/10 – Aula 8
19 e 20/10 – AVALIAÇÃO A1
24/10 – Aula 9
31/10 – Aula 10

NOVEMBRO

07/11 – Aula 11
14/11 – Aula 12
21/11 – Aula 13
28/11 – Aula 14

DEZEMBRO

04 a 08/12 – AVALIAÇÃO A3
11 e 12/12 – AVALIAÇÃO A2
19/12 – Término do semestre letivo

AVALIAÇÕES

A1 – Avaliação (30%)
A2 – Avaliação (30%)
A3 – Avaliação (40%)

<https://plataforma.bvirtual.com.br/Leitor/Publicacao/375/pdf/0?code=//KyNazKCKntXbLtSZ41HZHfjKxa41TtnKGSyJMHX8PaBEqY65EJsibjk8Bzi8dH6J2vegBMXFSKaOo04D4sQ==>

TECNOLOGIA EM DESTAQUE

O mundo corporativo migra para o código-fonte aberto

Sem contar com campanhas de propaganda multimídia, o sistema operacional de código-fonte aberto Linux já conquistou seu lugar no mundo corporativo. Ao oferecer mais flexibilidade e custos mais baixos do que o maior e dos produtos comerciais, o código aberto tem tudo para cair no gosto das empresas. Mas, como o movimento é novo e o cenário não está mapeado, trata-se de uma conversão deliberadamente cautelosa, e não de uma mudança imediata.

O benefício mais óbvio do código aberto é o acesso ao código-fonte que as empresas podem usar para integrar o Linux aos aplicativos empresariais existentes e aperfeiçoá-lo segundo seus próprios objetivos. Robert Leffowitz é vice-presidente de produtos e tecnologia executiva da Citicorp, uma empresa que usa Linux em suas operações. Leffowitz diz que o código-fonte aberto oferece a seus clientes, segundo ele, a mais fácil maneira de resolver problemas e manter a disponibilidade e a confiabilidade do sistema quando você pode "fazer uma cópia do código-fonte".

A Sigmatel, fabricante de placas, portais e equipamentos de ventilação, substituiu o sistema operacional Windows pelo Linux em seus servidores. Comprou o direito de usar o código-fonte de gerenciamento de estação de trabalho e de rede MySQL, implementou o código-fonte de dados MySQL e o servidor Web Apache, ambos de código aberto. Também manipulou o código-fonte do Linux, o que mais fácil para a empresa integrar seus aplicativos a um uso. Além disso, o Linux é mais confiável que o Windows. Quando usava o sistema da Microsoft, a empresa tinha de reiniciar seus servidores a cada duas semanas, porque o sistema operacional apresentava problemas. Agora, a Sigmatel diz que não precisa mais fazer isso.

Quem adotou o código aberto já verificou também outros benefícios. O estúdio cinematográfico DreamWorks Animation SPH economizou quatro meses de trabalho por PC, com software proprietário da Silicon Graphics, por PCs baseados em Linux. Com as economias de trabalho, a estúdio pagou algo entre 30 mil e 40 mil dólares por dia. De acordo com Jim Mariani, diretor de pesquisa e desenvolvimento, a migração para o Linux poucou ao redor 20 por cento no ano em despesas de licenciamento e operação. Mariani também acredita que o Linux tornou o código mais seguro contra ataques de vírus.

Em contrapartida a esses benefícios, as empresas e clientes precisam lidar com problemas e desafios que acompanham a incorporação do código aberto em uma infraestrutura de TI. Por exemplo, uma implementação bem-sucedida do código aberto exige manutenção e suporte apropriados. Se as empresas não tiverem acesso a recursos que possam oferecer tal suporte, as vantagens da inovação podem ser anuladas. Larry Kissel, CEO da Corelink, diz: "Não está sendo sempre apresentado entre as empresas como a Microsoft por não ter a vantagem de uma comunidade de código aberto onde a qual não tem controle".

Segundo Kamel Nassar, vice-presidente de estratégia de TI da Nielsen Media Research, a adoção do Linux em

suas empresas não avançou como deveria porque a base de conhecimento da equipe de TI era toda de Sun Solaris. Para Robert Kaito, o gerente-chefe de computação gráfica na Industrial Light & Magic, as empresas que estão migrando do UNIX para o Linux provavelmente terão melhores resultados do que aquelas que partem do Windows, porque os conhecimentos de suporte técnico exigidos pelo UNIX e pelo Linux são parecidos. A implementação de aplicativos de código aberto podem oferecer suporte, mas, se os clientes tiverem alterado o código e criado problemas com origem desconhecida, a ajuda será inútil.

D outro obstáculo para o uso do código aberto pelas empresas são os conhecimentos desatualizados e a desorganização das equipes que desenvolvem as aplicações. Quando uma empresa é grande, muitas vezes as equipes podem ser desorganizadas sem nenhum motivo, apenas porque os desenvolvedores não sabem que eles não devem mais falar.

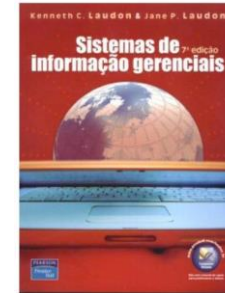
A estrutura mal definida da comunidade de desenvolvimento também expõe as empresas a problemas legais que se sajam entremetidos utilizando aplicativos comerciais. Quando as empresas fazem sentido a quem de propriedade intelectual que usam um produto desenvolvido, em certos casos, por milhares de pessoas ao redor do mundo, que podem reivindicar a propriedade. Acordos de licença para código aberto não são sempre suficientes para garantir a propriedade intelectual e a modificação do código. O National e o United Parcel Services (UPS), maiores usuários de servidores Linux, e a Redhat, a empresa que desenvolve o código-fonte do Linux, estão trabalhando para resolver esses problemas.

Além de avaliar as questões legais envolvidas no software de código aberto, empresas como a UPS e a ENTRADE não podem deixar a migração para código aberto e a utilização de software em um ambiente Linux sem cuidados. Verificar também se o código-fonte é realmente o código-fonte e se a implementação já existente. A maioria das empresas não se dá conta nem mais nem menos do software comercial que consome tantos investimentos. O Yahoo!, por exemplo, usa o código aberto para criar e aplicar os serviços mais usados pelos usuários, como e-mail e templates de páginas Web. No entanto, a empresa não pretende abandonar os aplicativos comerciais que processam a tecnologia de busca, a distribuição de clientes e os produtos online. A UPS, porém, espera processar todo o trabalho do site UPS com um servidor Linux até 2007.

Fonte: Larry Kissel, CEO da Corelink, em entrevista ao autor, 20 de maio de 2005; e Kamel Nassar, vice-presidente de estratégia de TI da Nielsen Media Research, 4 de junho de 2005.

Para passar:

Quais problemas o Linux e outros softwares de código aberto enfrentam as empresas a enfrentar? Como as ajudaram? Quais problemas e desafios fez aumentar a adoção de softwares de código aberto? O que pode ser feito para enfrentar esses problemas? Para você, qual seria a melhor estratégia para implantar o Linux e outros componentes de código aberto em seu atual estado de evolução?



1. Quais problemas o Linux e outros softwares de código aberto ajudaram as empresas a enfrentar?
2. Como as ajudaram?
3. Quais problemas e desafios fez aumentar a adoção de softwares de código aberto?
4. O que pode ser feito para enfrentar esses problemas?
5. Para você, qual seria a melhor estratégia para implantar o Linux e outros componentes de código aberto em seu atual estado de evolução?

<https://bellard.org/jslinux/>

JSLinux

Run Linux or other Operating Systems in your browser!

The following emulated systems are available:

CPU	OS	User Interface	VFsync access	Startup Link	TEMU Config	Comment
x86	Alpine Linux 3.12.0	Console	Yes	click here	url	
x86	Alpine Linux 3.12.0	X Window	Yes	click here	url	Right mouse button for the menu.
x86	Buildroot (Linux)	Console	Yes	click here	url	
x86	Buildroot (Linux)	X Window	Yes	click here	url	Right mouse button for the menu.
x86	Windows 2000	Graphical	No	click here	url	Disclaimer.
x86	FreeDOS	VGA Text	No	click here	url	
riscv64	Buildroot (Linux)	Console	Yes	click here	url	
riscv64	Buildroot (Linux)	X Window	Yes	click here	url	Right mouse button for the menu.
riscv64	Fedora 29 (Linux)	Console	Yes	click here	url	Warning: longer boot time.
riscv64	Fedora 29 (Linux)	X Window	Yes	click here	url	Warning: longer boot time. Right mouse button for

```

Loading...
Welcome to JS/Linux (i586)
Use 'vlogin username' to connect to your account.
You can create a new account at https://vfsync.org/signup .
Use 'export_file filename' to export a file to your computer.
Imported files are written to the home directory.

localhost:~# ls
bench.py  hello.c  hello.js  readme.txt
localhost:~# pwd
/root
localhost:~# █
  
```

© 2011-2020 Fabrice Bellard - [News](#) - [VM list](#) - [FAQ](#) - [Technical notes](#)

- Mostrar o **CPU** virtual que está sendo simulado pelo JSLinux com **1scpu**

```
localhost:~#  
localhost:~#  
localhost:~# lscpu  
Architecture:          i586  
CPU op-mode(s):        32-bit  
Byte Order:            Little Endian  
Address sizes:         32 bits physical, 32 bits virtual  
CPU(s):                1  
On-line CPU(s) list:   0  
Thread(s) per core:    1  
Core(s) per socket:    1  
Socket(s):             1  
Vendor ID:             AuthenticX86  
CPU family:            5  
Model:                 4  
Model name:            05/04  
Stepping:              3  
CPU MHz:               100.000  
BogoMIPS:              200.00  
Virtualization:        AMD-V  
Flags:                 fpu tsc msr pae cx8 cmov clflush mmx fxsr sse sse2 rdtscp cpuid svm npt  
localhost:~#
```

- Mostrar a **quantidade de memória** do sistema com **free**
 - Usar as opções **free -b** para mostrar em bytes, **free -k** em kiB, **free -m** em MiB

```
localhost:~#
localhost:~# free -b
              total        used        free      shared  buff/cache   available
Mem:      191737856     5021696    185765888         4096       950272    183549952
Swap:           0              0              0
localhost:~# free -k
              total        used        free      shared  buff/cache   available
Mem:       187244         4904       181412          4         928       179248
Swap:           0              0              0
localhost:~# free -m
              total        used        free      shared  buff/cache   available
Mem:         182           4         177          0           0         175
Swap:          0              0              0
localhost:~#
```

- Compilar um programa em C usando o `gcc` e executá-lo

```
localhost:~#  
localhost:~# cat > programa.c  
#include <stdio.h>  
  
float a, b, c;  
  
int main() {  
    a = 3.14;  
    b = 5.2;  
    c = a * b + 1;  
    printf("%f\n", c);  
    return 0;  
}  
localhost:~# gcc programa.c -o programa  
localhost:~# ./programa  
17.327999  
localhost:~#
```

Editores Linux

Vi programa.c

Editar = i

Sair e gravar = <ESC> :wq

OU

Nano programa.c

Isso vai gerar um arquivo executável (opção -o). Usar control-D para terminar o comando `cat`.

Ao executar, o resultado da operação é exibido no terminal.

- Mostrar o conteúdo do arquivo executável gerado (arquivo "programa") em hexadecimal usando `xxd`

```
localhost:~# xxd programa
00000000: 7f45 4c46 0101 0100 0000 0000 0000 0000  .ELF.....
00000010: 0300 0300 0100 0000 6810 0000 3400 0000  .....h...4...
00000020: 7c42 0000 0000 0000 3400 2000 0a00 2800  |B.....4. ...(.
00000030: 2100 2000 0600 0000 3400 0000 3400 0000  !. ....4...4...
00000040: 3400 0000 4001 0000 4001 0000 0400 0000  4...@...@.....
00000050: 0400 0000 0300 0000 7401 0000 7401 0000  .....t...t...
00000060: 7401 0000 1700 0000 1700 0000 0400 0000  t.....
00000070: 0100 0000 0100 0000 0000 0000 0000 0000  .....
00000080: 0000 0000 6003 0000 6003 0000 0400 0000  .... ..
00000090: 0010 0000 0100 0000 0010 0000 0010 0000  .....
000000a0: 0010 0000 cd02 0000 cd02 0000 0500 0000  .....
000000b0: 0010 0000 0100 0000 0020 0000 0020 0000  .....
000000c0: 0020 0000 b800 0000 b800 0000 0400 0000  . ....
000000d0: 0010 0000 0100 0000 fc2e 0000 fc3e 0000  .....>..
000000e0: fc3e 0000 0801 0000 3401 0000 0600 0000  .>.....4.....
000000f0: 0010 0000 0200 0000 0c2f 0000 0c3f 0000  ...../...?..
00000100: 0c3f 0000 c000 0000 c000 0000 0600 0000  .?.....
00000110: 0400 0000 50e5 7464 0c20 0000 0c20 0000  ....P.td. ...
00000120: 0c20 0000 2400 0000 2400 0000 0400 0000  . ..$...$.....
00000130: 0400 0000 51e5 7464 0000 0000 0000 0000  ....Q.td.....
00000140: 0000 0000 0000 0000 0000 0000 0600 0000  .....
00000150: 1000 0000 52e5 7464 fc2e 0000 fc3e 0000  ....R.td.....>..
00000160: fc3e 0000 0401 0000 0401 0000 0400 0000  .>.....
```


- Mostrar o arquivo executável decodificado em instruções de máquina usando

```
objdump -d
```

```
localhost:~#
localhost:~# objdump -d programa

programa:      file format elf32-i386

Disassembly of section .init:

00001000 <_init>:
 1000:      83 ec 0c                sub    $0xc,%esp
 1003:      e8 be 01 00 00        call   11c6 <frame_dummy>
 1008:      e8 8d 02 00 00        call   129a <__do_global_ctors_aux>
 100d:      83 c4 0c                add    $0xc,%esp
 1010:      c3                     ret

Disassembly of section .plt:

00001020 <.plt>:
 1020:      ff b3 04 00 00 00      pushl  0x4(%ebx)
 1026:      ff a3 08 00 00 00      jmp     *0x8(%ebx)
 102c:      00 00                  add     %al,(%eax)
  ...

00001030 <printf@plt>:
 1030:      ff a3 0c 00 00 00      jmp     *0xc(%ebx)
 1036:      68 00 00 00 00        push    $0x0
 103b:      e9 e0 ff ff ff        jmp     1020 <.plt>

00001040 <__libc_start_main@plt>:
 1040:      ff a3 10 00 00 00      jmp     *0x10(%ebx)
 1046:      68 08 00 00 00        push    $0x8
 104b:      e9 d0 ff ff ff        jmp     1020 <.plt>

Disassembly of section .plt.got:
```

- Mostrar no código desassemblado as instruções que fazem os cálculos que aparecem no programa fonte (multiplicação e adição de ponto flutuante). Mostrar que as instruções são executadas em uma sequência de endereços de memória crescentes.

```

1210:      c3                ret
00001211 <__x86.get_pc_thunk.cx>:
1211:      8b 0c 24          mov     (%esp), %ecx
1214:      c3                ret
00001215 <main>:
1215:      8d 4c 24 04       lea     0x4(%esp), %ecx
1219:      83 e4 f0          and     $0xffffffff0, %esp
121c:      ff 71 fc          pushl  -0x4(%ecx)
121f:      55                push   %ebp
1220:      89 e5             mov     %esp, %ebp
1222:      53                push   %ebx
1223:      51                push   %ecx
1224:      e8 e4 ff ff ff    call    120d <__x86.get_pc_thunk.ax>
1229:      05 a3 2d 00 00     add     $0x2da3, %eax
122e:      8d 90 60 00 00 00  lea     0x60(%eax), %edx
1234:      d9 80 38 e0 ff ff  flds    -0x1fc8(%eax)
123a:      d9 1a             fstps   (%edx)
123c:      8d 90 58 00 00 00  lea     0x58(%eax), %edx
1242:      d9 80 3c e0 ff ff  flds    -0x1fc4(%eax)
1248:      d9 1a             fstps   (%edx)
124a:      8d 90 60 00 00 00  lea     0x60(%eax), %edx
1250:      d9 02             flds    (%edx)
1252:      8d 90 58 00 00 00  lea     0x58(%eax), %edx
1258:      d9 02             flds    (%edx)
125a:      de c9             fmulp   %st, %st(1)
125c:      d9 e8             rcpd    %st, %st(1)
125e:      de c1             faddp   %st, %st(1)
1260:      8d 90 5c 00 00 00  lea     0x5c(%eax), %edx
1266:      d9 1a             fstps   (%edx)
1268:      8d 90 5c 00 00 00  lea     0x5c(%eax), %edx
126e:      d9 02             flds    (%edx)
1270:      83 ec 04          sub     $0x4, %esp

```

Exercícios Linux

1. Criar um Shell Script

Vi oi.sh

```
#!/bin/bash
echo "Oi, tudo bem?"
~
~
```

Digitar os comandos “:wq” → Enter

Executar os comandos abaixo:

```
localhost:/professor# chmod a+x Oi.sh
localhost:/professor# ls
Oi.sh
localhost:/professor# ls -l
total 4
-rwxr-xr-x  1 root    root      34 Aug 24 22:28 Oi.sh
localhost:/professor# ./Oi.sh
Oi, tudo bem?
```

Editores Linux

Vi programa.c

Editar = i

Sair e gravar = <ESC> :wq

OU

Nano programa.c

2. Criar o Shell Script abaixo:

Primeiro shell script

```
#!/bin/bash
```

```
echo "Hello, World!"
```



```
TEXTO="Hello, World!"
```

```
echo $TEXTO
```



```
#!/bin/bash
data=$(date +"%y/%m/%d-%H%M")
echo $1 "$data"-$2
```

Exemplo de loop e condição

```
#!/bin/bash
```

```
COUNT=0
```

```
for i in `ls $1`
```

```
do
```

```
done
```

```
COUNT=$(( if [ "$COUNT" -eq 0 ]; then
```

```
echo "Nenhum arquivo encontrado"
```

```
elif [ "$COUNT" -eq 1 ]; then
```

```
echo "Apenas 1 arquivo foi encontrado"
```

```
else
```

```
echo "Foram encontrados $COUNT arquivos"
```

```
fi
```

Exercícios

1. Determine o número que se obtém ao se escrever o número 3 no sistema de numeração de base 2.

A) (111)

B) (101)

C) (10)

D) (01)

E) (11)

Temos que $3 = 1 \cdot 2 + 1 \cdot 1$

Logo, 3 na base 2 é 11

2. Considere $A = (11000)$ e $B = (10001)$, números escritos no sistema de numeração de base 2. Escreva-os no sistema de numeração de base 10 e determine o valor de $A - B$.

- A) -7
- B) 41
- C) -17
- D) 0
- E) 7

Passando para a base 10:

$$11000 = 1 \cdot 16 + 1 \cdot 8 + 0 \cdot 4 + 0 \cdot 2 + 0 \cdot 1 = 24$$

$$10001 = 1 \cdot 16 + 0 \cdot 8 + 0 \cdot 4 + 0 \cdot 2 + 1 \cdot 1 = 17$$

$$24 - 17 = 7$$

3. Converter $110111_{(2)}$ para a base decimal.

$$110111_{(2)} = 1x2^0 + 1x2^1 + 1x2^2 + 0x2^3 + 1x2^4 + 1x2^5 = 55_{(10)}$$

4. Converter $1101100_{(2)}$ para a base hexadecimal.

Como 16 é múltiplo de 2 (2^4), basta separar o número dado de 4 em 4 algarismos da direita para a esquerda e convertê-los da base 2 para a base 10 para se encontrar o resultado na base 16.

$$1101100_{(2)} = 110|1100 = \underline{6C}_{(16)}$$

5. Converter $198_{(10)}$ para a base binária.

$$\begin{aligned}198/2 &= 99, \text{ resto } 0 \\99/2 &= 49, \text{ resto } 1 \\49/2 &= 24, \text{ resto } 1 \\24/2 &= 12, \text{ resto } 0 \\12/2 &= 6, \text{ resto } 0 \\6/2 &= 3, \text{ resto } 0 \\3/2 &= 1, \text{ resto } 1 \\198_{(10)} &= 11000110_{(2)}\end{aligned}$$

6. Converter $889_{(10)}$ para a base hexadecimal.

7. Converter $ABB_{(16)}$ para a base binária.

8. Converter $FEA_{(16)}$ para a base decimal.

9. Converter o número binário 10011011 para decimal:

10. Conversão do Binário 100101100 para Hexadecimal :

Obrigado