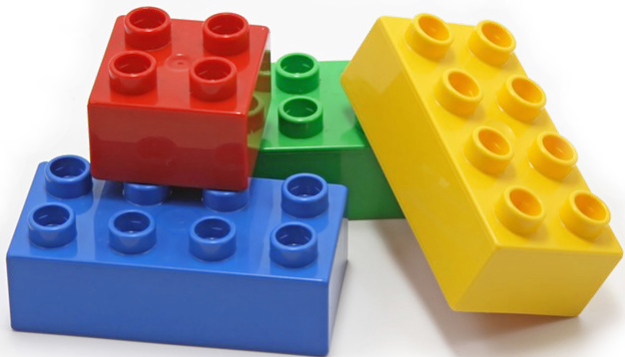


Universidade Federal do Amazonas
Instituto de Computação
DevTITANS - Projeto de Desenvolvimento, Tecnologia e Inovação em
Android e Sistemas Embarcados



DevTITANS



Metodologia

▣ Aulas Teóricas

- Visão geral do conteúdo
- 30 a 45 minutos de teoria
- Rápidos exemplos práticos

▣ Laboratórios Práticos

- Estilo tutorial
- Complementam o conteúdo teórico
- Laboratórios duram entre 30min e 1h
 - *A ideia é fazer 1 ou 2 por dia*
 - *Se não der tempo de terminar, não tem problema*
 - *Pode ser feito após o término da aula ou na próxima semana*
- Professor estará disponível para dúvidas

Avaliação



- ▣ Laboratórios de Programação (LP)
 - ▣ 1 a 2 por aula
- ▣ Presença

Laboratórios Práticos

- ▣ Todos os laboratórios são **individuais**
 - ▣ Pode discutir e tirar dúvidas entre si, mas sem cópia de código
- ▣ Acesso a um Sistema Linux
 - ▣ Os laboratórios consideram que você possui acesso a um sistema Linux
 - *Máquinas do laboratório*
 - *Laptop/PC próprio*

Moodle



- ▣ Moodle
 - ▣ Será usado para **disponibilizar o material** da disciplina e os laboratórios
 - ▣ devtitans.icomp.ufam.edu.br/moodle

- ▣ Sua conta
 - ▣ Login e Senha enviados por e-mail

Dúvidas

- Dúvidas poderão ser tiradas por **E-Mail**:
 - Professor: yfa@icomp.ufam.edu.br
 - Monitor: mop@icomp.ufam.edu.br
- Videoconferência
 - Qualquer aluno pode solicitar um atendimento via videoconferência
 - Dia e horário a combinar

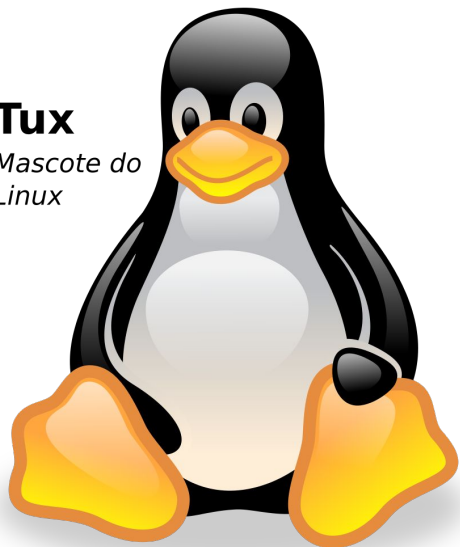
Universidade Federal do Amazonas

Instituto de Computação

DevTITANS - Projeto de Desenvolvimento, Tecnologia e Inovação em
Android e Sistemas Embarcados

Tux

*Mascote do
Linux*



Introdução ao Linux



Assunto



- ▣ Faremos uma breve introdução ao sistema operacional Linux
 - Por enquanto, será focado mais no Linux e menos no AOSP
 - Futuramente, veremos como o Linux está relacionado com o Android e o AOSP
- ▣ Nivelar os conhecimentos em Linux
 - Alunos com diferentes níveis de experiência com o sistema
- ▣ Reforçar e focar apenas no necessário ao AOSP
 - Não veremos todo o conteúdo de Linux (e.g., GUI, Servidores)
 - Focaremos no que será necessário nos próximos módulos

Sobre o Linux

- ▣ Software livre – 4 liberdades
 - Liberdade de **executar** o programa, para qualquer propósito
 - Liberdade de **estudar** o programa, e adaptá-lo para as suas necessidades
 - Liberdade de **modificar** (aperfeiçoar) o programa e distribuir estas modificações, de modo que toda a comunidade se beneficie
 - Liberdade de **redistribuir** cópias do programa de modo que você possa ajudar ao seu próximo

Sobre o Linux

- ▣ Código fonte aberto
 - ▣ Licença GPL (*GNU General Public License*)

- ▣ GPL - *GNU General Public License*
 - ▣ Provê as quatro liberdades do software livre
 - ▣ Entretanto, possui uma restrição:
 - Licença “copyleft”, ou seja, um software que é uma modificação de outro que usa a GPL, só pode ser distribuído se utilizar a mesma licença
 - ▣ Outras licenças não possuem essa restrição e são ainda mais livres
 - BSD (*Berkeley Source Distribution*)



Sobre o Linux

Rocky Racoon,
Mascote do Minix



- ▣ Criado em 1991 por Linus Torvalds
 - ▣ Divulgado pela primeira vez em um fórum Usenet chamado `comp.os.minix`:

Hello everybody out there using **minix** -

I'm doing a (free) operating system (just a **hobby**, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things).

I've currently ported **bash**(1.08) and gcc(1.40), and things seem to work. This implies that I'll get something practical within a few months, and I'd like to know what features most people would want. Any suggestions are welcome, but I won't promise I'll implement them :-)

Linus (torvalds@kruuna.helsinki.fi)

Sobre o Linux

- ▣ O Linux pode ser usado principalmente como:
 - ▣ Sistema **desktop** tradicional
 - ▣ Sistema para **servidores**
 - ▣ Sistema para **dispositivos** embarcados
- ▣ Focaremos mais no último caso
 - ▣ Desenvolvimento e personalização do Android em sistemas embarcados

Linux no Desktop

- ▣ O Linux possui programas correspondentes à maioria dos programas usados em outros sistemas:
 - ▣ Navegador: **Chrome, Firefox**
 - ▣ Office: **Google Docs, Sheets e Slides**
 - ▣ Edição de Fotos: **Krita, Gimp**
 - ▣ Edição de Imagens Vetoriais: **Inkscape**
 - ▣ Edição de Vídeos: **Kdenlive, Olive Editor**
 - ▣ Desenvolvimento: **Kdevelop, Eclipse, IntelliJ, Sublime, Visual Code Studio**

Linux no Servidor

- Talvez, uma das maiores aplicações do Linux seja nos servidores
- Provavelmente, todos os principais serviços online que você usa estão hospedados em um servidor Linux
 - Google
 - Facebook
 - Amazon
 - Wikipedia
 - Twitter



Linux em Dispositivos Embarcados

- Smartphones (Android)
- Tablets, eReaders
- Relógios
- TVs, Consoles (Jogos)
- Roteadores
- Carros
- Outros Dispositivos:
 - Raspberry Pi
 - Beagle Board



Distribuições Linux

- ▣ Para facilitar o uso do Linux, este é usado através de *distribuições*
- ▣ Uma distribuição contém não apenas o kernel Linux, mas uma série de **outros programas** que permitem o uso do mesmo.
- ▣ As distribuições mais conhecidas hoje em dia são:
 - ▣ Debian
 - ▣ Ubuntu
 - ▣ Fedora
 - ▣ Mint



Desktop Environments

- O Linux em si é apenas o *kernel* (núcleo) do sistema.

- A interface gráfica que vemos é implementada por um gerenciador de janelas conhecido como X.Org



- Já o desktop é conhecido como *desktop environment*. O Linux possui vários *desktop environments*:

- GNOME
- KDE
- Cinnamon
- Xfce



Instalação



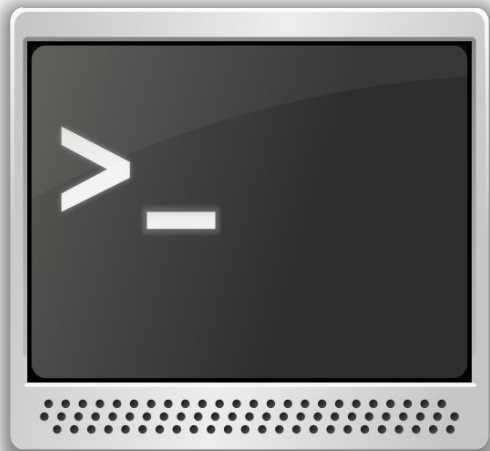
- ▣ Não iremos entrar nos detalhes da instalação de uma distribuição
 - Varia de uma distribuição para outra
 - Entre no site da distribuição para ver os detalhes de cada uma

- ▣ Entretanto, os **passos principais** e comuns entre eles são:
 - Entrar no site da distribuição
 - Baixar o arquivo ISO da versão mais recente
 - Criar um pendrive USB de boot
 - Reiniciar o Laptop/PC a partir do pendrive
 - Seguir os passos de instalação que, geralmente, envolvem:
 - *Detecção e configuração de hardware*
 - *Selecionar o esquema de particionamento do disco*
 - *Cópia dos arquivos*
 - *Reboot*

Universidade Federal do Amazonas

Instituto de Computação

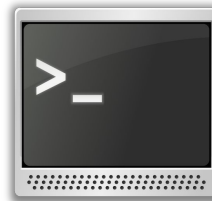
DevTITANS - Projeto de Desenvolvimento, Tecnologia e Inovação em
Android e Sistemas Embarcados



Sobre o Terminal



Sobre o Terminal



- ▣ O terminal é uma **interface texto** para o sistema
 - ▣ Em contraste com uma interface gráfica
- ▣ É também conhecido como:
 - ▣ **Shell**, *console*, linha de comando, ou *prompt*
- ▣ Em geral, pessoas mais experientes com o Linux costumam usar muito o terminal
 - ▣ Apesar de hoje em dia não ser mais tão necessário
 - ▣ O terminal acaba se mostrando como uma forma mais prática e rápida de fazer as coisas, para quem já tem experiência nele

Terminal Neste Curso

- ▣ Neste e nos próximos módulos, tudo será feito no terminal
- ▣ Por ser em modo texto:
 - ▣ O terminal é o ambiente ideal para seguir instruções de tutoriais e laboratórios
 - ▣ Permite copiar e colar textos e comandos
 - ▣ Muitos dos comandos usados pelo AOSP não possuem correspondentes gráficos
- ▣ Se você não possui experiência com o terminal, não se preocupe!
 - ▣ A ideia é você ir aprendendo e se familiarizando à medida que você for fazendo os laboratórios

Executando Comandos

```
$ ls
```

Diretório Atual

\$ pwd

Listando Arquivos



```
$ ls /etc/host*  
$ ls -l /bin/ls
```


Listando Arquivos

-rw-r--r--	1	root	root	42	Oct 21 2015	/etc/hosts
Tipo	Links	Dono	Grupo	Tamanho	Data de Modificação	Nome do Arquivo
Permissões						

Listando Arquivos



```
$ ls -lh /bin/ls  
$ ls -al
```

Manual dos Comandos

```
$ man ls  
$ tldr ls
```

Navegando em Diretórios



```
$ cd /var/log  
$ pwd
```

Indo para o Home



\$ cd

Voltando para o Dir. Anterior

```
$ cd -  
$ pwd
```

Diretórios . e ..



```
$ ls -a
```

```
$ cd .
```

```
$ cd ..
```

Universidade Federal do Amazonas

Instituto de Computação

DevTITANS - Projeto de Desenvolvimento, Tecnologia e Inovação em
Android e Sistemas Embarcados

```
cc_binary {  
  name:    "hello_c",  
  vendor:  true,  
  srcs:    [ "hello_c.c" ],  
  shared_libs: [ "liblog" ],  
}
```

Android.bp

Trabalhando com Arquivos



Lendo Arquivos



```
$ cat /etc/hostname  
$ cat /etc/lsb-release
```

Lendo Arquivos



```
$ cat /etc/login.defs | wc -l  
$ more /etc/login.defs
```

Lendo Arquivos

```
$ head -5 /etc/passwd
```

Lendo Arquivos



```
$ tail -5 /etc/passwd
```

Editando Arquivos



```
$ vi ArquivoTeste.txt
```

Editando Arquivos



```
$ nano ArquivoTeste.txt
```

Copiando Arquivos



```
$ cp ArquivoTeste.txt ArquivoCopia.txt
```

Movendo Arquivos



```
$ mv ArquivoCopia.txt /tmp  
$ mv ArquivoTeste.txt ArquivoRenomeado.txt
```

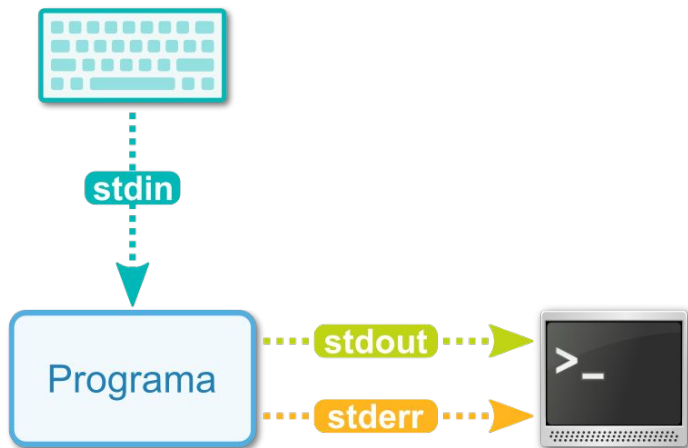

Diretórios

```
$ mkdir TesteDir  
$ ls -d */  
$ rmdir TesteDir
```

Universidade Federal do Amazonas

Instituto de Computação

DevTITANS - Projeto de Desenvolvimento, Tecnologia e Inovação em
Android e Sistemas Embarcados



Entrada e Saída Padrões, Redirecionamento, Pipes



Entrada e Saída Padrões

- ▣ As mensagens normais dos programas em modo texto são enviadas para o que é conhecido como **saída padrão**
 - ▣ *stdout - standard output*
 - ▣ Normalmente, essa saída padrão é o próprio terminal
- ▣ Além da saída padrão, tem-se também a saída de erro
 - ▣ *stderr - standard error*
 - ▣ Normalmente, a saída de erro é o terminal também
- ▣ Por fim, os programas possuem também uma entrada padrão
 - ▣ Em geral, essa entrada é o teclado
 - ▣ Nem todos os programas esperam alguma entrada, mas os que precisam, o fazem através da entrada padrão

Entrada e Saída Padrões

- A figura a seguir mostra uma visão geral dessas entradas e saídas:



- As saídas podem ser redirecionadas para:
 - Arquivos (próximo slide)
 - Programas (mais adiante)

Redirecionando para Arqs.



```
$ ls -lh /usr/bin > ArquivosDoBin.txt  
$ echo "Linha1" > ArquivoTeste.txt  
$ echo "Linha2" >> ArquivoTeste.txt
```

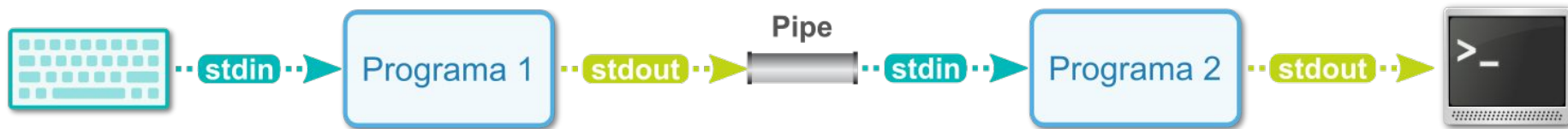
Redireccionando Erros



```
$ ls ArchivoInvalido.txt 2> lsStderr.txt  
$ ls ArchivoInvalido.txt 2> /dev/null
```

Redirecionando para Programas

- A saída de um programa pode ser redirecionada e usada como entrada por outro programa
- Isso é feito através do caractere “|” (traço vertical)
 - Conhecido como *pipe* (do inglês, cano, tubo)



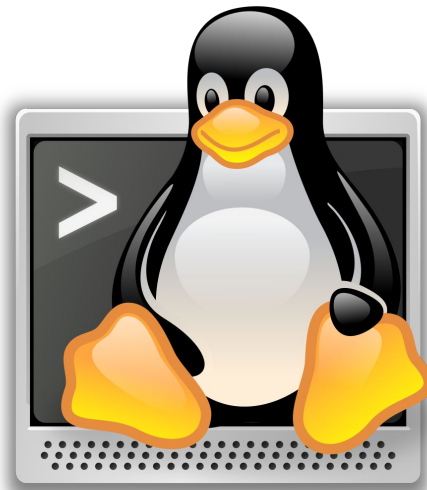
Pipe



```
$ cat /proc/cpuinfo | grep "model name"
```


Laboratório!

- ▣ Entre no Moodle:
 - ▣ devtitans.icomp.ufam.br/moodle
- ▣ Faça os laboratórios:
 - ▣ Laboratório 1: Introdução ao Linux, Linha de Comando e Arquivos
 - ▣ Laboratório 2: Entrada e Saída Padrões, Redirecionamento, Pipes



Universidade Federal do Amazonas
Instituto de Computação
DevTITANS - Projeto de Desenvolvimento, Tecnologia e Inovação em
Android e Sistemas Embarcados



Sistema de Arquivos

Introdução



Hierarquia de Diretórios



- ▣ No Linux, os diretórios são representados pelo símbolo /
 - ▣ / *Forward Slash* (barra), usado no Linux
 - ▣ \ *Backslash* (barra invertida), usado no Windows
- ▣ Um sistema Linux é composto por diversos diretórios
 - ▣ Cada um com objetivos bem determinados.

Hierarquia de Diretórios

- ▣ Separação baseado no tipo/objetivo dos arquivos:
 - Sistema bem organizado
 - Fácil de se manter
 - *Arquivos de configurações de todos os programas ficam no diretório /etc, permitindo encontrá-los rapidamente*
 - *Arquivos executáveis ficam no /bin, permitindo que estes sejam rapidamente encontrados ao serem executados*
 - *Bibliotecas compartilhadas ficam no /lib, permitindo que elas sejam facilmente encontrados por todos os programas que precisam delas*

Pacotes

- ▣ Uma desvantagem deste tipo de organização é que os arquivos de um único programa ficam espalhados pelo sistema
- ▣ Solução: **pacotes**
 - ▣ Uma forma padronizada de se distribuir programas
 - ▣ Possui instruções de instalação, dependências, configuração e desinstalação do programa
 - ▣ Praticamente todos os arquivos do Linux fazem parte de algum pacote
 - ▣ O sistema de instalação inicial do Linux nada mais é do que a instalação de todos os pacotes necessários para um sistema básico
 - *Iremos falar mais da instalação de pacotes futuramente.*

Discos, Partições e Montagem

- ▣ No Linux, não existe o conceito de "*drives*" (e.g., C:\)
- ▣ Todos os discos/partições são "montados" em um diretório
 - E não em "letras" como no Windows
 - O diretório raiz (/) é uma partição, enquanto que outros subdiretórios, como o /home ou /boot, podem ser outras partições/discos
- ▣ **Montar** uma partição significa que estamos mapeando um disco/partição para um diretório.

Partições

- ▣ Um disco pode ser dividido em **partições**
 - ▣ Para o sistema operacional, cada partição se parece com um disco diferente
 - ▣ No Windows, é comum termos apenas uma única partição
 - ▣ Já no Linux/Unix, é comum termos várias partições.
- ▣ Por exemplo, pode-se ter partições **diferentes** para:
 - ▣ / Sistema principal
 - ▣ /boot Permite o sistema iniciar em recuperação mesmo se o / ficar cheio
 - ▣ /home Sistema continua normal, mesmo com o home cheio
- ▣ Partições possuem um **tipo** (sistema de arquivos)
 - ▣ ext4, ntfs, vfat

Diretório Raiz (/)



\$ 1s /

Diretórios do Sistema

Dir.	Descrição
/etc	Arquivos de configurações
/bin,	Arquivos binários (executáveis) normais
/sbin	Arquivos binários essenciais
/lib	Bibliotecas compartilhadas
/boot	Contém os arquivos principais de inicialização (kernel, grub)
/home	Diretórios dos usuários
/root	Diretório do usuário root
/dev	Arquivos de acesso a dispositivos de hardware
proc, /sys	Arquivos de comunicação com módulos e componentes do kernel
/media, /mnt	Diretório para montagem de outras partições (mnt) e pendrives (media)
/tmp	Arquivos temporários
/var	Contém arquivos "variáveis", que mudam com frequência (log, cache)
/usr	Arquivos de programas de usuários, menos relacionados ao sistema

Explorando o /etc

```
$ ls /etc
```

Explorando o /etc



```
$ ls -lh /etc/passwd /etc/shadow /etc/group
```

Explorando o /etc



```
$ cat /etc/fstab
```

Explorando o /etc



```
$ cat /etc/hostname
```

Explorando o /etc



```
$ cat /etc/timezone
```

Explorando o /dev

\$ ls /dev

Explorando o /dev



```
$ cat /dev/random  
$ echo "msg" > /dev/stdout
```


Explorando o /proc



```
$ ls /proc
```

Explorando o /proc



```
$ cat /proc/loadavg
```

Explorando o /proc



```
$ cat /proc/uptime  
$ uptime
```

Explorando o /bin

```
$ ls /bin
```

Explorando o /bin



```
$ hexdump -C /bin/ls | head -3
```

Explorando o /bin



```
$ /bin/date ou simplesmente date  
$ echo $PATH
```

Explorando o /bin



```
$ df -h
```

Explorando o /bin



```
$ who
```


Universidade Federal do Amazonas

Instituto de Computação

DevTITANS - Projeto de Desenvolvimento, Tecnologia e Inovação em
Android e Sistemas Embarcados



Sistema de Arquivos

Comandos, Links

Físicos e Simbólicos



Path dos Binários



\$ `whereis` who

Path dos Binários



```
$ whereis hexdump
```

Buscando Arquivos



```
$ find /etc  
$ find /etc 2> /dev/null | wc -l
```

Buscando Arquivos



```
$ find /usr/include | grep stdio
```

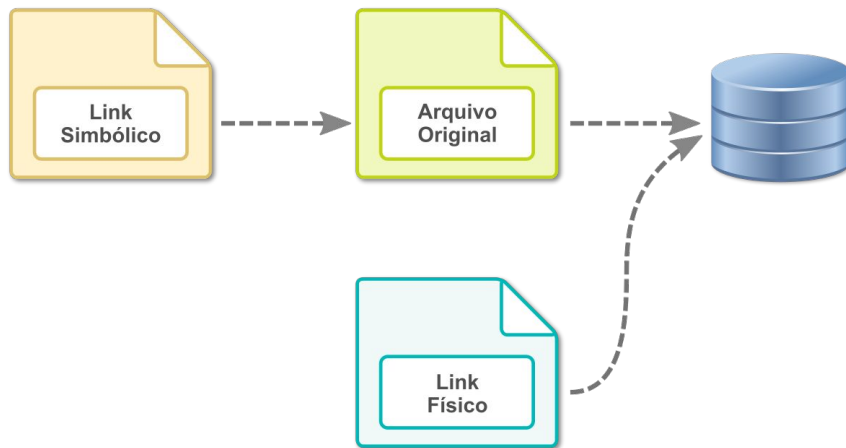
Comando *Disk Usage*



```
$ cd /usr  
$ du -sh  
$ du -sh *
```

Links Simbólicos e Físicos

- Alguns arquivos são, na verdade, links (referências, ligações) para outros arquivos
 - Links são criados através do comando *ln*
- Existem dois tipos de links:
 - Links simbólicos
 - Links físicos



Listando Arquivos

-rw-r--r--	1	root	root	42	Oct 21 2015	/etc/hosts
Tipo	Links	Dono	Grupo	Tamanho	Data de Modificação	Nome do Arquivo
Permissões						

Link Simbólico

```
$ echo "Mensagem arq. original" > ArquivoOriginal.txt  
$ ln -s ArquivoOriginal.txt ArquivoLinkSimbolico.txt  
$ ls -lh ArquivoOriginal.txt ArquivoLinkSimbolico.txt
```

Link Simbólico



```
$ echo "Msg. Concatenada" >> ArchivoLinkSimbolico.txt  
$ cat ArchivoOriginal.txt
```

Link Simbólico



```
$ rm ArquivoOriginal.txt  
$ ls -lh ArquivoLinkSimbolico.txt
```

Link Físico



```
$ echo "Mensagem original 2" > ArquivoOriginal2.txt  
$ ln ArquivoOriginal2.txt ArquivoLinkFisico.txt  
$ ls -lh ArquivoOriginal2.txt ArquivoLinkFisico.txt
```

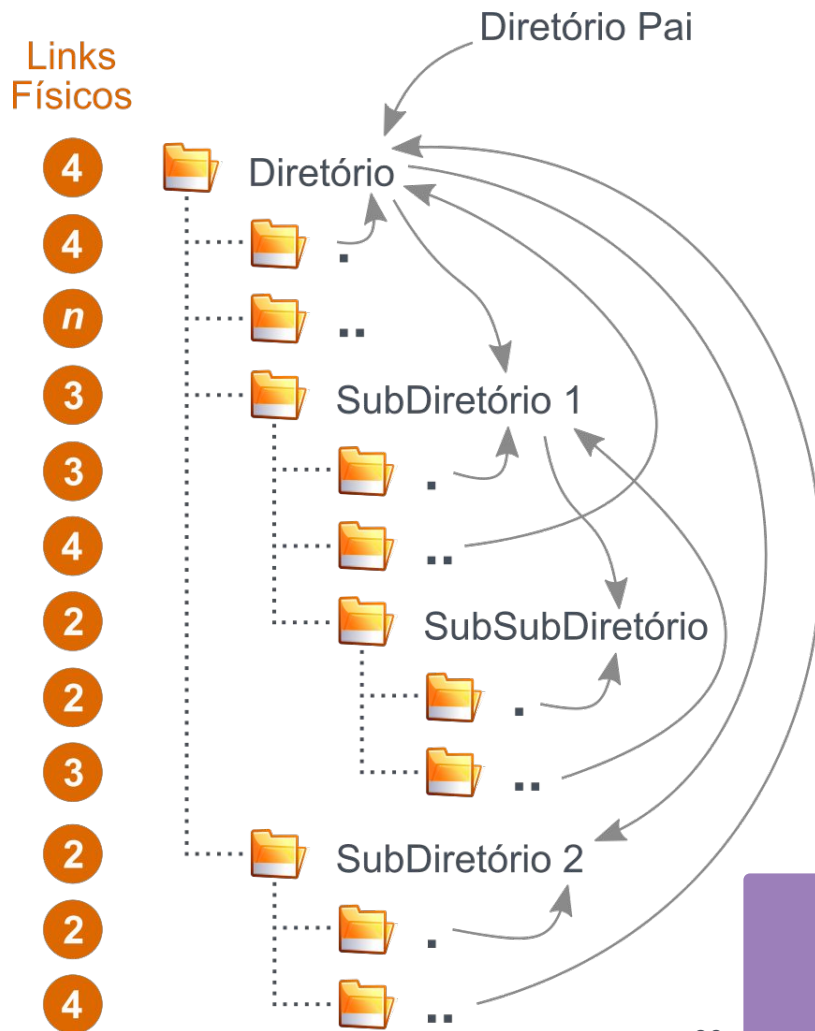
Link Físico



```
$ rm ArquivoOriginal2.txt  
$ ls -lh ArquivoLinkFisico.txt
```

Diretórios . e ..

- Os diretórios . e .. apontam para o diretório atual e anterior, respectivamente.
- Para “apontar” pra tais diretórios, são usados **links físicos**



Diretórios . e ..



```
$ ls -alh | grep "\." | head -2  
$ mkdir NovoSubdiretorio
```

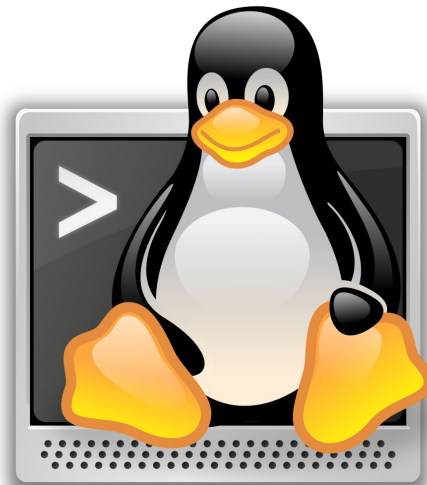
Laboratório!

- Entre no Moodle:

- devtitans.icomp.ufam.br/moodle

- Faça os laboratórios:

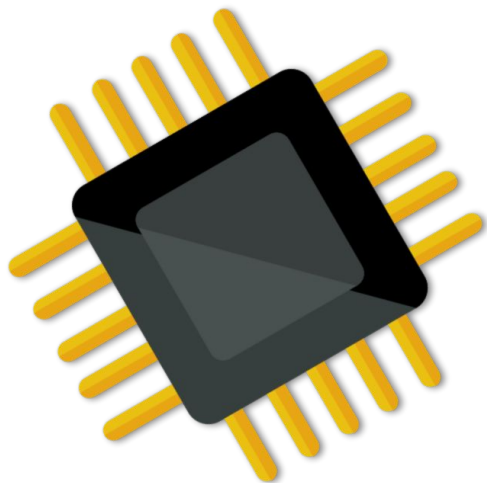
- Laboratório 3: Sistema de Arquivos - Explorando
 - Laboratório 4: Sistema de Arquivos - Comandos, Links Físicos e Simbólicos



Universidade Federal do Amazonas

Instituto de Computação

DevTITANS - Projeto de Desenvolvimento, Tecnologia e Inovação em
Android e Sistemas Embarcados



Processos, Daemons, Serviços, Inicialização do Sistema



Processos



- ▣ Um processo é um programa em execução
- ▣ Em um sistema multitarefa, centenas de processos estão rodando ao mesmo tempo e usando os recursos do processador
 - ▣ O tempo de CPU é dividido entre os processos pelo sistema operacional através do seu escalonador de processos.

PID e PPID

- ▣ Todo processo possui um número único no sistema, conhecido como PID (Process Identifier)
 - ▣ Este número é parâmetro para vários comandos relacionados a processos que iremos ver nesta aula
- ▣ Todo processo possui um pai, conhecido como PPID (Parent Process Identifier)
 - ▣ Quando você está no terminal usando o interpretador de comandos bash e pede para executar um comando, o pai desse novo processo será o próprio bash

Listando Processos

```
$ ps  
$ cat &  
$ ps
```

Listando Processos



```
$ ps -e  
$ ps -ef  
$ ps -ef | wc -l
```

Processos e Recursos



```
$ top  
$ top -o %MEM
```

Matando um Processo



```
$ ps | grep cat  
$ kill <PID>  
$ kill -9 <PID>
```

Matando um Processo pelo Nome



```
$ cat &  
$ cat &  
$ killall -9 cat
```


Memória Livre



```
$ free  
$ free -h
```

Daemons



- ▣ São processos que proveem um serviço para outros processos
 - ▣ devido a isso, estão sempre em execução
- ▣ Os daemons são iniciados no *boot* do sistema e só terminam quando o sistema é desligado
- ▣ O Linux possui vários processos *daemons*

Daemons

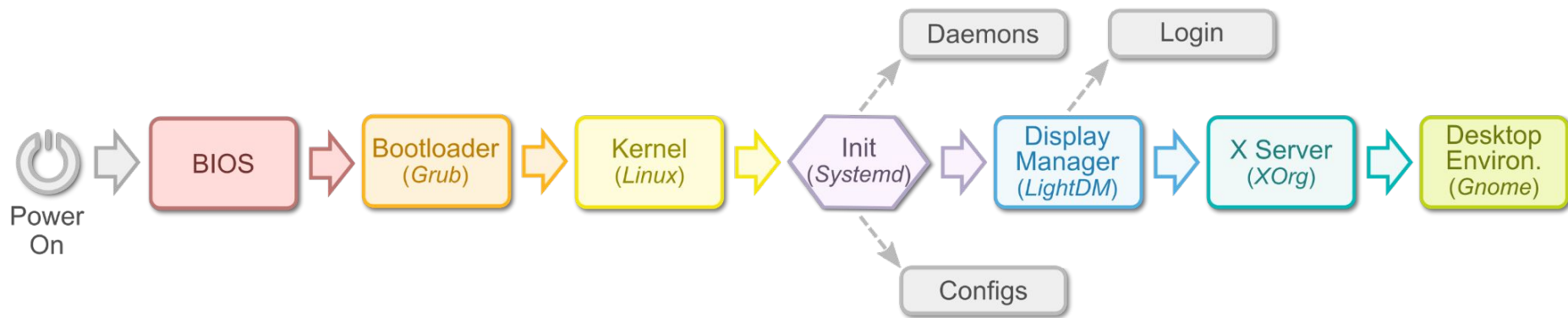
<i>Daemon</i>	Descrição
systemd-udev	Monitora mensagens de eventos de hardware do kernel
systemd-resolved	Responsável por "resolver" (descobrir o IP) de um nome DNS (<i>Domain Name System</i>)
systemd-timesyncd	Mantém o relógio do computador sincronizado com um servidor NTP (<i>Network Time Protocol</i>)
systemd-logind	Gerencia o login e as sessões de usuários
cron	Permite o agendamento de comandos rotineiramente (todos os dias, todas as horas, etc)
atd	Permite o agendamento de comandos para um dia/horário específico
rsyslogd	Provê suporte às mensagens de log (do diretório /var/log)
thermald	Monitora a temperatura da CPU e previne o aquecimento do sistema
acpid	Monitora eventos de hardware e notifica usuários (laptop aberto, bateria, etc)
cupsd	Servidor de impressão do Linux
Xorg	Servidor X. Oferece os serviços básicos de interface gráfica.

Daemons



```
$ ps -ef | grep systemd
```

Inicialização do Linux



Init



```
$ ps -ef | grep init  
$ ls -lh /sbin/init  
$ ls /lib/systemd/system/cron.service
```

Universidade Federal do Amazonas

Instituto de Computação

DevTITANS - Projeto de Desenvolvimento, Tecnologia e Inovação em
Android e Sistemas Embarcados



Segurança no Linux, Usuários, Senhas, Permissões



Segurança no Linux



- ▣ O Linux utiliza duas formas principais de segurança:
 - ▣ Discretionary Access Control (DAC)
 - ▣ Mandatory Access Control (MAC)
- ▣ O **Discretionary Access Control (DAC)** se baseia no uso de login, senhas e permissões de acesso a arquivos, diretórios e processos
 - ▣ É a forma mais comum e conhecida de segurança no Linux e em outros sistemas

Segurança no Linux

- ❑ O **Mandatory Access Control (MAC)** se baseia em limitar o acesso de programas a um conjunto mínimo de recursos
 - ❑ Recursos: arquivos, outros processos, memória, portas TCP, variáveis, etc
 - ❑ Bloqueando todo acesso a recursos não explicitamente permitido
 - ❑ As duas principais implementações no Linux é o
 - *SELinux (usado pelo Android)*
 - *AppArmor (usado no Ubuntu e no Mint).*
- ❑ Vamos focar no Discretionary Access Control (DAC), pois o MAC será visto melhor no Android.

Atenção ao usar o Root

- ▣ Nos laboratórios relacionados a segurança, alguns comandos serão executados como root (administrador)
 - ▣ Através do comando **sudo**
- ▣ Apesar de ser difícil de um usuário normal tirar um Linux do ar, é extremamente fácil para o root executar um comando errado e danificar o sistema
 - ▣ O Linux confia cegamente e executa todos os comandos do root, sem verificar se eles estão corretos ou analisar se eles podem prejudicar o sistema



Usuários



```
$ tail -5 /etc/passwd  
$ head -1 /etc/passwd  
$ cat /etc/passwd | grep sshd
```

Formato do /etc/passwd

booker : x : 1004 : 2000 : Booker DeWitt : /home/booker : /bin/bash

Login Password User Id Group Id Name Home Dir Shell

Criando um Usuário



```
$ sudo adduser <LoginEscolhido>
```

Senhas



```
$ passwd
```

```
$ sudo cat /etc/shadow
```

Formato do /etc/shadow

fulano:\$6\$ECLRnIt6\$THR2PuQdikirMPpv...BYhKrHvp.ZZ7G/1:19012:0:99999:7:20:39378:

Login Encrypted Password Last Change Minimum Maximum Warn Inactive Expire

\$6\$ECLRnIt6\$THR2PuQdikirMPpv...BYhKrHvp.ZZ7G/1

Type Salt Hash

Grupos



```
$ cat /etc/group
```


Formato do /etc/group

`adm : x : 4 : syslog, valentina, jebediah`

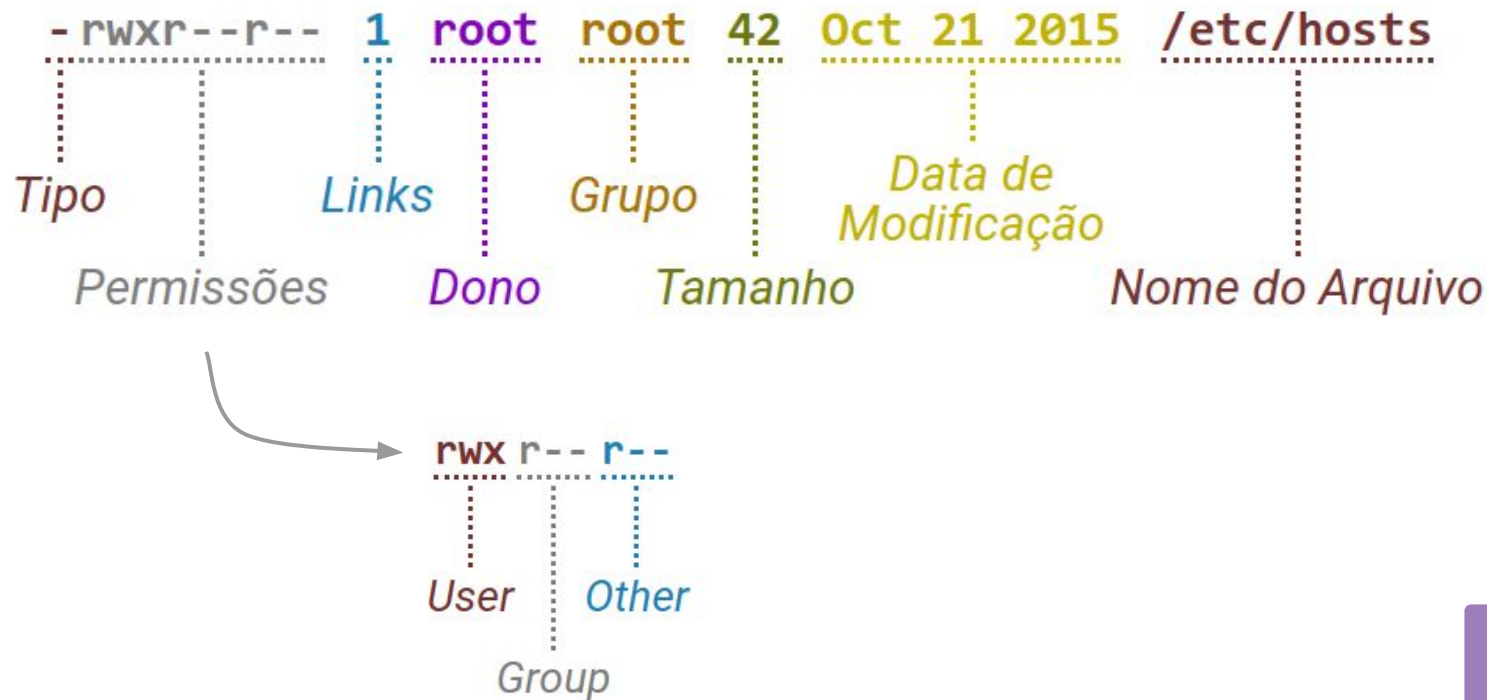
Name *Password* *ID* *List of Users*

Criando Grupos



```
$ sudo addgroup devtitans  
$ sudo usermod -a -G devtitans horacio
```

Permissões dos Arquivos



Permissões



```
$ echo '#!/bin/bash' > HelloBash.sh
$ echo 'echo "Hello Bash ..."' >> HelloBash.sh
$ ls -l HelloBash.sh
```

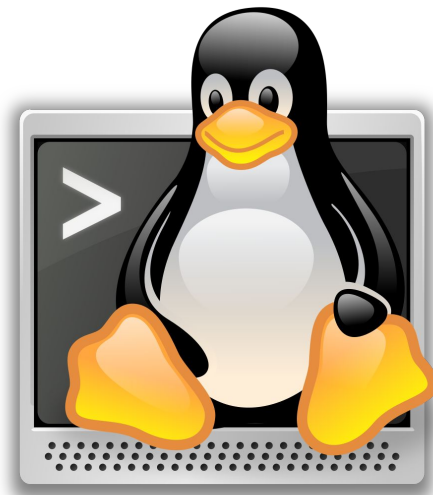
Permissões



```
$ ./HelloBash.sh  
$ chmod u+x HelloBash.sh  
$ ./HelloBash.sh
```

Laboratório!

- ▣ Instale o PuTTY (cliente SSH):
 - ▣ bit.ly/putty-baixar
- ▣ Na janela "PuTTY Configuration"
 - ▣ Host Name: tardis.icomp.ufam.edu.br
 - ▣ Port: 8080
 - ▣ Clique em "Open" -> "Yes"
- ▣ Será pedido o login e a senha:
 - ▣ Login: iniciais do seu nome (Ex: yfa)
 - ▣ Senha: Titan@<iniciais>123 (Ex: Titan@yfa123)
- ▣ Por fim, entre no Moodle:
 - ▣ <https://bit.ly/devtitans-moodle>



Universidade Federal do Amazonas
Instituto de Computação
DevTITANS - Projeto de Desenvolvimento, Tecnologia e Inovação em
Android e Sistemas Embarcados



Tux
*Mascote do
Linux*

Introdução ao Kernel do Linux



Introdução



- ▣ Desenvolvido por Linus Torvalds (1991)
- ▣ Baseado no Unix
- ▣ Escrito nas linguagens de programação C e Assembly
- ▣ Democrático, então todos podem contribuir
- ▣ O kernel Linux está licenciado sob a GNU General Public License v2
- ▣ O android é baseado no Kernel do Linux

Kernel do Linux



- ▣ Principais recursos
 - Portabilidade: funciona na maioria das arquiteturas.
 - Escalabilidade: poucos MB de RAM é suficiente.
 - Segurança: não pode esconder suas falhas pois seu código é revisado.
 - Modularidade: pode incluir apenas o necessário do sistema.
 - Fácil de programar: você pode aprender com código existente.

- ▣ O carregador de inicialização faz o que é necessário para extrair e descompactar o kernel na RAM e depois passar o controle para ele

Inicialização do Kernel



```
$ ls -l /boot
```

Espaço do Kernel

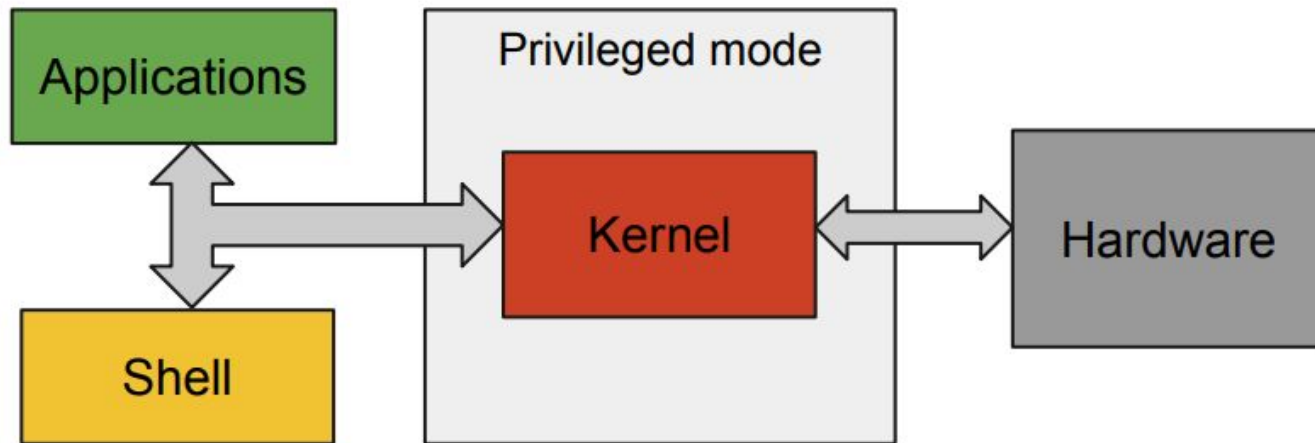
- ▣ O kernel encontra-se em um estado de sistema elevado
 - ▣ Espaço de memória protegido.
 - ▣ Acesso total ao hardware do dispositivo.
- ▣ O acesso principal ao hardware e aos serviços do sistema são gerenciados e fornecidos como um serviço para o restante do sistema

Espaço do Usuário



- ▣ As aplicações do usuário são realizadas no espaço do usuário
 - ▣ Jogo.
 - ▣ Software de produtividade.
 - ▣ Aplicativos normais.
- ▣ Um subconjunto dos recursos disponíveis da máquina por meio de chamadas de sistema do kernel
 - ▣ Não possui acesso direto ao hardware.

Fluxo de Comunicação



Gerenciamento de Recursos



- ▣ Interface de chamada do sistema (SCI)
 - Uma camada fina que fornece um método para interagir de espaço do usuário para o espaço do kernel.

- ▣ Gestão de Processos (PM)
 - Criar, destruir processos.
 - Comunicação entre diferentes processos.

- ▣ Gerenciamento de memória (MM)
 - Gerenciamento de memória física para virtual.
 - Alocação de memória.

Gerenciamento de Recursos



- ▣ Sistemas de arquivos
 - Funcionalidades de sistemas de arquivo.

- ▣ Drivers de dispositivo (DD)
 - Interagir com o hardware.
 - Extrair uma abstração das funcionalidades do dispositivo.

- ▣ Pilha de rede
 - Implementação dos protocolos de rede.
 - Entregar pacotes entre programas e interfaces de rede.

Tipos de Núcleo

▣ Monolítico

- ▣ As versões anteriores do kernel Linux eram de tal forma que todas as suas partes eram fixadas estaticamente em um único bloco gigante.

▣ Modulares

- ▣ O kernel moderno têm a maior parte de suas funcionalidades contidas em módulos que são colocados no kernel dinamicamente.

Núcleo Modular



▣ Vantagens

- ▣ Facilitam o desenvolvimento de drivers sem reinicialização: carregar, testar, descarregar, reconstruir.
- ▣ Útil para manter o tamanho da imagem do kernel no mínimo.
- ▣ Útil para reduzir o tempo de inicialização.
- ▣ Possibilidade de permitir apenas módulos, ou desabilitar totalmente o suporte ao módulo.

Versão do Kernel



```
$ uname -r  
$ cat /proc/version
```

Configuração de Compilação



```
$ cat /boot/config-$(uname -r)
```

Pseudo Sistemas de Arquivos

- ▣ Linux disponibiliza informações do sistema e do kernel no espaço do usuário por meio de pseudo sistemas de arquivos
- ▣ Permitem que aplicativos vejam diretórios e arquivos que não existem no armazenamento real
- ▣ São criados e atualizados dinamicamente pelo kernel

Pseudo Sistemas de Arquivos

Os dois sistemas de pseudo-arquivos mais importantes são:

- `/proc`
 - Informações relacionadas ao sistema operacional.
 - Processos.
 - Gerenciamento de memória.

- `/sys`
 - Representação do sistema como uma árvore de dispositivos conectados.
 - Frameworks do kernel que gerenciam esses dispositivos.

CPU



```
$ cat /proc/cpuinfo
```

```
$ cat /proc/cpuinfo | grep processor
```

Memória



```
$ cat /proc/meminfo  
$ cat /proc/meminfo | grep MemTotal
```

Partições



```
$ cat /proc/partitions
```


Barramentos



```
$ lspci | tail -5
```

USB



```
$ lsusb | tail -5  
$ lsusb -v
```

Variáveis do Kernel



```
$ sysctl -a  
$ cat /proc/sys/net/ipv4/tcp_fin_timeout
```

Dispositivos Conectados

\$ ls /dev

Universidade Federal do Amazonas
Instituto de Computação
DevTITANS - Projeto de Desenvolvimento, Tecnologia e Inovação em
Android e Sistemas Embarcados



Linux Loadable Kernel Module (LKM)



LKM's



- ▣ Oferecem uma maneira fácil de estender a funcionalidade do kernel
 - ▣ A maioria dos drivers são implementados como módulos de kernel.
 - ▣ Podemos descarregar apenas esse driver específicos.
 - ▣ Permitem que o kernel se comunique com o hardware sem precisar saber como o hardware funciona.

LKM's

- ▣ Cada peça é chamada de "módulo"
- ▣ Pode ser carregado em tempo de execução
- ▣ Amplia a funcionalidade do sistema
- ▣ Não há necessidade de reconstruir o kernel
- ▣ Pode economizar memória (carregar apenas o necessário)

Listar Módulos

```
$ lsmod  
$ ls /sys/module
```


Character Device Driver

- Hardwares são acessados pelo usuário por meio de arquivos de dispositivos especiais.
- Geralmente permite apenas acesso sequencial byte a byte.
- Implementa: abrir, fechar, ler, escrever.
- Semelhante a arquivos normais.
- Exemplos:
 - Teclado
 - Mouse
 - Porta Seriais
 - Placa de Som

Mensagem de Log



\$ dmesg

Programa de Usuário



- ▣ Um programa geralmente começa com uma função `main()`, executa um monte de instruções e termina após a conclusão das instruções.

- ▣ Os programas de usuários seguem o fluxo:
 - ▣ Alocação de memória para o programa.
 - ▣ Carregamento de quaisquer bibliotecas compartilhadas necessárias.
 - ▣ Execução da instrução começa a partir em algum ponto (normalmente o `main`).
 - ▣ Exceções são lançadas, a memória dinâmica é alocada e desalocada e o programa eventualmente é executado até a conclusão.

Módulo do Kernel

- Não executa sequencialmente
 - Um módulo do kernel se registra para lidar com solicitações usando sua função de inicialização, que é executada e termina com a função de remoção.
- Não tem limpeza automática
 - Quaisquer recursos alocados ao módulo devem ser liberados manualmente quando o módulo é descarregado.
- Não tem funções de usuário como o `printf()`
 - O código do kernel não pode acessar bibliotecas de código que são escritas para o espaço do usuário Linux.
 - Tem uma função `printk()` que pode gerar informações, que podem ser visualizadas no espaço do usuário.

Módulo Hello World



```
$ cd /devtitans-kernel  
$ cat newlkm-devtitans.c
```

Makefile



- ▣ Faz a maior parte da construção para um desenvolvedor.
- ▣ Inicia o sistema de construção do kernel e fornece ao kernel informações sobre os componentes necessários para construir o módulo.

Makefile - Compilar

```
$ nano Makefile  
$ make
```

Listar Arquivos

\$ 1s

Informações do Módulos



\$ `modinfo newlkm-devtitans.ko`

Inserir Módulo



```
$ sudo insmod newlkm-devtitans.ko
```

Listar Módulo



```
$ lsmod | grep newlkm_devtitans
```

Log de Criação

\$ dmesg

Remover Módulo



```
$ sudo rmmmod newlk-devtitans
```

Log de Remoção

\$ dmesg

Inserir Parâmetro



```
$ sudo insmod newlkm-devtitans.ko name=DevTitans
```

Mensagem de Log

\$ dmesg

Listar Módulo



```
$ cat /proc/modules | grep newlkm_devtitans
```

Listar Dispositivo



```
$ ls -l /sys/module/newlkm_devtitans
```

Detalhes de Parâmetro



```
$ cd parameters/  
$ ls -l  
$ cat name
```

Remover Módulo



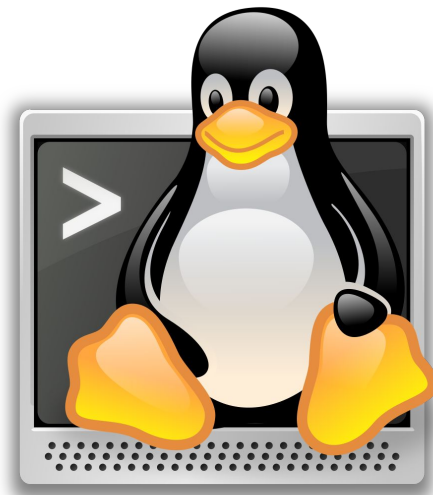
```
$ sudo rmmmod newlk-devtitans
```

Log de Remoção

\$ dmesg

Laboratório de Kernel!

- Instale o PuTTY (cliente SSH):
 - bit.ly/putty-baixar
- Na janela "PuTTY Configuration"
 - Host Name: tardis.icomp.ufam.edu.br
 - Port: 8080
 - Clique em "Open" -> "Yes"
- Será pedido o login e a senha:
 - Login: iniciais do seu nome (Ex: yfa)
 - Senha: Titan@<iniciais>123 (Ex: Titan@yfa123)
- Por fim, entre no Moodle:
 - <https://bit.ly/devtitans-moodle>



Universidade Federal do Amazonas

Instituto de Computação

DevTITANS - Projeto de Desenvolvimento, Tecnologia e Inovação em
Android e Sistemas Embarcados



Introdução ao Git e GitHub



Sobre o Git



- Foi desenvolvido pela comunidade que mantém o kernel do Linux, tendo Linus Torvalds como principal desenvolvedor.
- A maior parte das operações do Git são executadas na máquina local do usuário.
 - Dispensa o uso de um servidor central
 - Isso significa que há poucas coisas que você não possa fazer caso esteja offline
- Não é GITHUB.

Sobre o Git



- Pode usá-lo para acompanhar arquivos de texto ou até imagens. Isso significa que o Git não é apenas para desenvolvedores.
- Alguns casos de uso:
 - Manter um histórico das versões anteriores
 - Desenvolvimento simultâneo em diferentes ramos
 - Experimentar facilmente novos recursos, integrá-los na produção ou descartá-los.
 - Colaboração com outros desenvolvedores
- Um sistema de controle de versão distribuído que permite rastrear as alterações feitas em seus arquivos ao longo do tempo.

Instalação

- ▣ Para usar o Git você precisa instalá-lo em seu computador.
 - ▣ Fazer o download para o seu sistema operacional a partir das opções fornecidas no site (<https://git-scm.com/downloads>)
 - ▣ Linux: apt-get install git-core
 - ▣ Mac: <http://code.google.com/p/git-osx-installer/>
- ▣ Git bash, extensões do eclipse como eGit e GUI do GitHub.
- ▣ Utilizar o comando **git --version** para verificar se o git já está instalado.

Configuração

- ▣ O Git vem com uma ferramenta chamada *git config* que permite a definição de variáveis de configuração.
- ▣ Este procedimento é necessário para o Git saber sua identificação:
 - ▣ `git config --global user.name "Yuri Assayag"`
 - ▣ `git config --global user.email "yfa@icomp.ufam.edu.br"`
- ▣ Quando executar qualquer comando, o Git fará o reconhecimento por estas informações.

Repositório

- ▣ Um repositório é uma pasta onde ficarão armazenados todos os arquivos de um projeto.
- ▣ Qualquer pessoa que possua uma cópia do repositório pode acessar toda a base de código e seu histórico.
- ▣ Permite:
 - ▣ Interação com o histórico
 - ▣ Clonagem
 - ▣ Criação de ramificações
 - ▣ Mesclagem
 - ▣ Comparação de alterações entre versões de código e muito mais.

Iniciando um Repositório

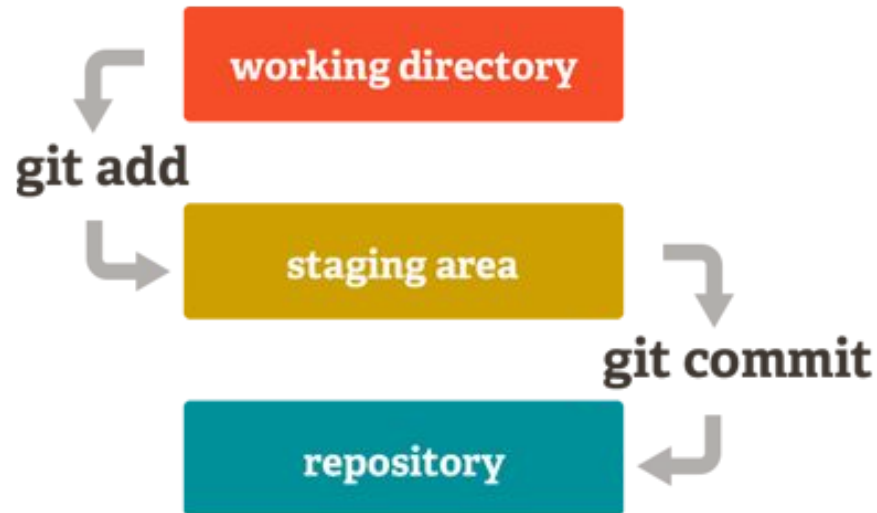
- ▣ **git init** – cria projeto git no diretório existente.
 - Faz o Git começar a “observar” as mudanças no diretório
 - Adiciona uma subpasta oculta dentro do diretório existente que abriga a estrutura de dados necessária para o controle de versão
- ▣ Comandos para criar um novo repositório local:
 - `$ mkdir devtitans-git`
 - `$ cd devtitans-git`
 - `$ git init`
 - `$ ls -la`

Visualizando Diretório

```
$ cd .git
```

Fluxo de Trabalho

- O fluxo de trabalho básico:
 - Selecciona os arquivos que deseja versionar (**git add**)
 - Consolida tais arquivos no repositório (**git commit**)



Fluxo de Trabalho pt1

- ▣ Working Directory
 - Quando os arquivos são criados na pasta, mas o arquivo é novo, ele se encontra no estágio de Untracked Files
 - Isso acontece porque ele é um arquivo desconhecido pelo git
 - Untracked files são as alterações pendentes que nós precisamos adicionar na staging area

Criando Novo Arquivo



```
$ echo "print('Hello Git')" >>  
hello.py
```

Status

Working Directory

```
$ git status
```

Fluxo de Trabalho pt2

▣ Staging Area

- ▣ É como o Git acompanha as alterações que você deseja que sejam inseridas em seu próximo commit
- ▣ Rodamos o comando **git add** para mover um novo arquivo para a área de teste
- ▣ A área de teste reflete o conteúdo exato do arquivo quando você executou o git add

Adicionando na Staging Area



```
$ git add hello.py
```

Status

Staging Area

```
$ git status
```



Fluxo de Trabalho pt3

- ▣ Repositório (committed)
 - Após o arquivo ser adicionado a staging area, o arquivo está pronto para ser commitado.
 - Quando falamos commitados, estamos querendo dizer que este arquivo fará parte, efetivamente, do repositório
 - Crie um commit dos arquivos adicionados com o comando **git commit**, seguido por uma mensagem de commit
 - Use git status para ver o status atual da árvore de trabalho

Comitando Novo Arquivo



```
$ git commit -m  
"novo commit"
```

Status

Comitted

```
$ git status
```



Alterando um Arquivo

- ▣ Mudanças no arquivo
 - Modificar o conteúdo de “hello.py”
 - Verificar o status
 - Usar o git add hello.py
 - Verificar o status
 - Usar o git commit -m “novo commit no hello.py”

Adicionando Multiplos Archivos

```
$ git add .  
$ git commit -m ...
```

Git Log

- ▣ O log do Git mostra o histórico dos commits.
 - ▣ Todas as mensagens de commit são mostradas em ordem
 - ▣ O início de cada commit é marcado com a palavra “commit” seguida pelo SHA daquele commit
 - ▣ É uma ferramenta extremamente importante e poderosa para utilizar.
- ▣ Tipos de log:
 - ▣ `git log -p`
 - ▣ `git log -p -2`
 - ▣ `git log --pretty=oneline`
 - ▣ `git log --pretty=format:"%h - %an, %ar : %s"`

O Arquivo .gitignore

- ▣ Especifica os arquivos que você não deseja que o Git rastreie.
- ▣ Comumente usado para:
 - ▣ Arquivos compilados
 - ▣ Binários
 - ▣ Grandes arquivos (imagens)
- ▣ Tenha cuidado se você adicionar um arquivo ao .gitignore depois de já ter sido rastreados.

.gitignore



```
$ touch arquivoIgnorado.txt  
$ git status  
$ nano .gitignore
```

.gitignore



```
$ git status  
$ git add .gitignore  
$ git commit -m ...
```

Voltando Versões

- ▣ Uma das maneiras mais simples é utilizando o comando checkout, passando o hash do commit.
 - ▣ Consultamos os commits feitos e buscamos a hash do commit que queremos voltar (*git log --pretty=oneline*)
 - ▣ O estado atual aponta para o commit que informamos no comando checkout

Removendo um Arquivo

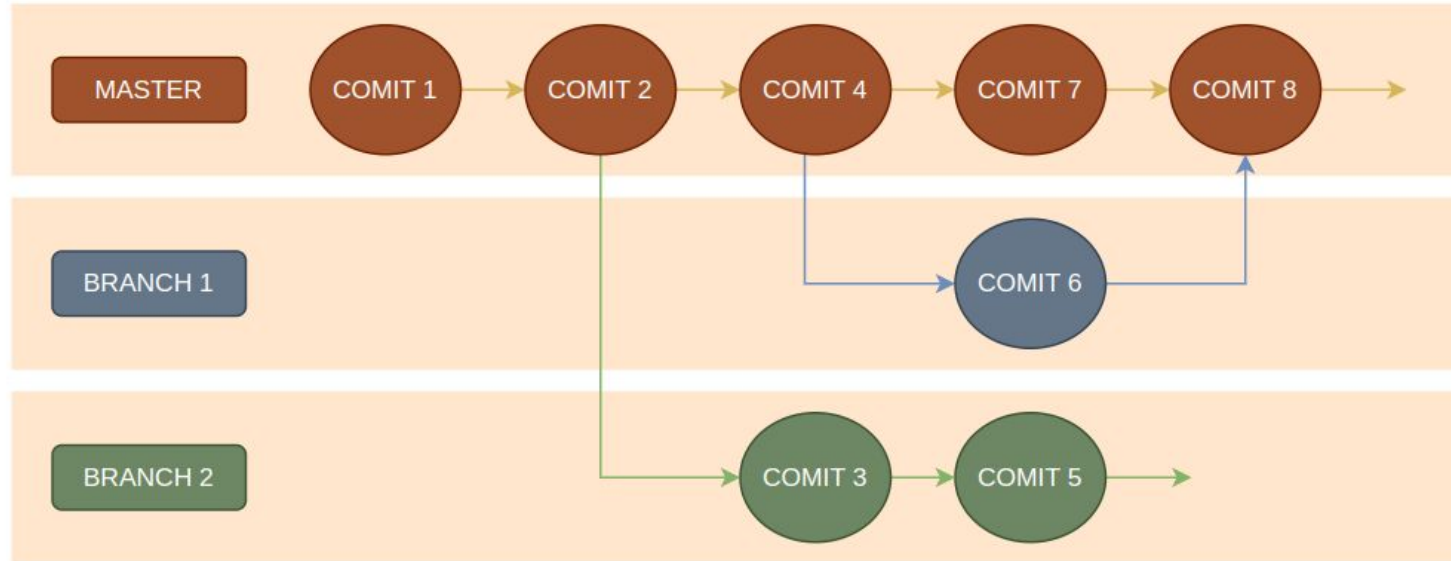


- ▣ Para remover um arquivo da árvore de trabalho e no próximo commit, execute **git rm <filename>**
- ▣ Para removê-lo do próximo commit, mas manter o arquivo na árvore de trabalho, execute **git rm --cached <filename>**

Falando sobre Branch

- ▣ Usado principalmente para criar um novo recurso, sem se preocupar com alterações no projeto principal.
 - ▣ O git trabalha como se fosse uma linha do tempo e esta linha seria o branch principal (master)
- ▣ Permite trabalhar em um mesmo projeto de forma assíncrona, de modo que, muitos desenvolvedores possam trabalhar em um mesmo projeto, sem um atrapalhar o outro.
- ▣ Criar um branch é fazer um novo commit, contendo todos os commits anteriores a ele.

Branch



Principais Comandos

- ▣ Principais comandos para uso de branch:
 - Listar todas as ramificações do projeto – **git branch**
 - Criar um novo branch – **git branch <branchname>**
 - Mudar para um branch – **git checkout <branchname>**
 - Criar e alternar imediatamente – **git checkout -b <branchname>**
 - Excluir uma ramificação – **git branch -d <branchname>**

Visualizando Branch



```
$ git branch
```

Criando Novo Branch



```
$ git checkout -b novafuncionalidade  
$ git branch
```

Adicionando Archivos

```
$ touch funcionalidade1.py  
$ git add funcionalidade1.py  
$ git commit -m ...
```

Voltar Branch



```
$ git checkout master  
$ ls  
$ git log --pretty=oneline
```

Merge



- ▣ Este comando é normalmente usado para combinar alterações feitas em duas branches distintas.
 - ▣ Um desenvolvedor usa o merge quando quiser combinar as alterações de recurso em uma branch específica na branch mestre para implantação final de uma nova funcionalidade
- ▣ O Git cria um novo commit que combina os principais SHAs de duas ramificações, se necessário.

Merge Conflitos

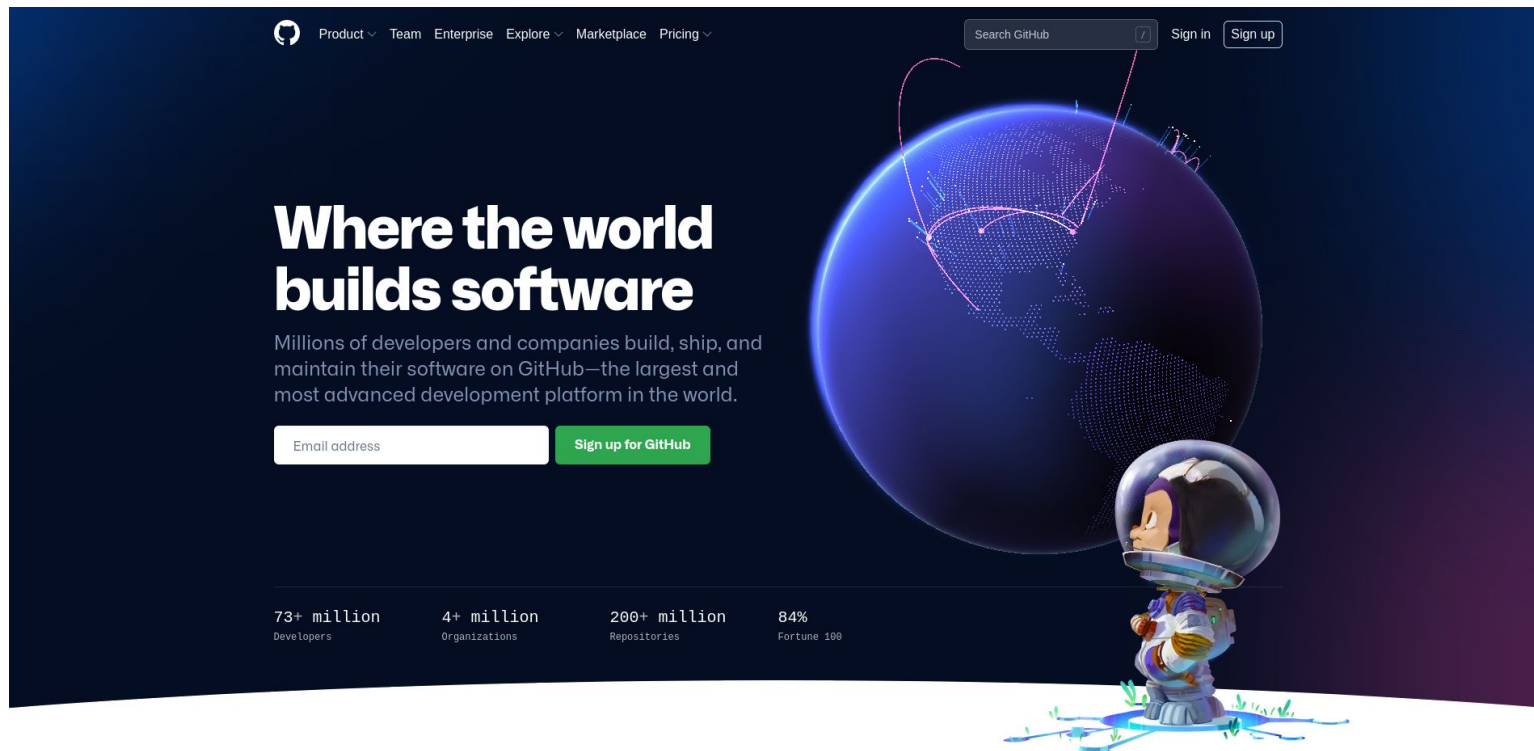
- ▣ Às vezes, duas ramificações editarão o mesmo pedaço de código de maneiras diferentes.
- ▣ Deve-se resolver o conflito manualmente e, em seguida, adicionar o arquivos conflitantes e confirme explicitamente.

Fazendo Merge



```
$ git merge novafuncionalidade  
$ ls
```

Sobre o GitHub



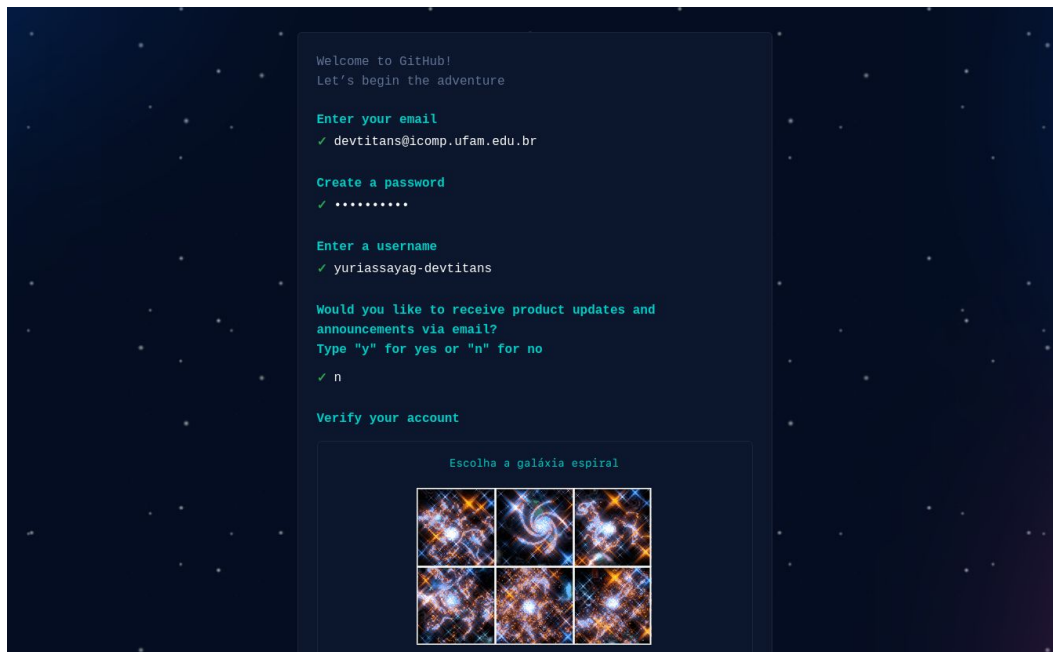
Sobre o GitHub



- É um site que hospeda repositórios git em um servidor remoto e nos auxilia a gerenciar tais repositórios.
- Pode acessar e baixar um projeto em qualquer computador ao qual tenha acesso.
 - Hospedar repositórios no Github facilita o compartilhamento de bases de código entre as equipes, fornecendo uma GUI para facilmente clonar repositórios para uma máquina local
- Permite fazer suas alterações e enviar a versão mais recente de volta ao repositório online.

Cadastro

- Criar uma conta no GitHub (<https://github.com/>), que é um cadastro básico como em qualquer site ou serviço.




Criando Novo Repositório

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner *

 yuriassayag ▾

Repository name *

Great repository names are short and memorable. Need inspiration? How about [bug-free-octo-meme?](#)

Description (optional)

☒  **Public**

Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

☐ **Add a README file**

This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore

Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: None ▾

Choose a license

A license tells others what they can and can't do with your code. [Learn more.](#)

License: None ▾

 You are creating a public repository in your personal account.

Create repository

Push



- ▣ O push, em inglês, é “empurrar”.
- ▣ É utilizado quando temos um repositório local e queremos “empurrar” os commits para um repositório online.
- ▣ É necessário informarmos o endereço do repositório online, para que, toda vez que dermos um push, o Git saiba para onde enviar os arquivos.

Push



- ▣ Para criar este link e configurar o repositório local, é necessário executar o comando:
 - ▣ `git remote add origin <link_do_repositorio>`
- ▣ Após configurar o link do servidor remoto é possível utilizar o push para enviar o repositório local para o github:
 - ▣ `git push origin master`

Autenticação



- ▣ Atualmente o GitHub pede uma autenticação por token para se comunicar com o repositório remoto.
- ▣ Para a criação do Token é necessário executar os seguintes passos:
 - Acessar sua conta do GitHub e clique em **Settings**
 - Ir em **Developer Settings**, à esquerda
 - Clique em **Personal Access Tokens** e a seguir em **Generate New Token**
 - Adiciona uma pequena nota sobre o token a ser gerado e marcar “repo”
 - Escolha a data de expiração do token (opcional) e clique em **Generate Token**
 - É importante guardar o número do token gerado

Push



```
$ git push origin master
```

Push em Novo Branch



```
$ git checkout novafuncionalidade  
$ git push origin novafuncionalidade
```

Clone



- ▣ O comando clone serve para baixar o repositório online para a máquina local.
- ▣ Através do repositório remoto é possível obter o link para clonagem via https ou ssh:
 - ▣ Via https: `git clone <link_do_repositorio>`
 - ▣ Via ssh: `git@github.com:username/repositorio.git`

Remover Repositório Local



```
$ cd ..
```

```
$ rm -rf devtitans-git
```

Clonando Repositório



```
$ git clone <link> <pasta>
```

Pull



- ▣ Usado para buscar e baixar conteúdo de repositórios remotos e fazer a atualização imediata ao repositório local para que os conteúdos sejam iguais.
- ▣ O que faz é atualizar suas branches locais de acordo com as branches remotas.

Pull



```
$ git pull
```


Trabalhando com Push e Pull

▣ Parte 1

- Faça um novo clone do repositório com o nome “clone 2”
- No repositório “devtitans-git” vá para o branch master
- Modifique o arquivo hello.py, adicione e dê commit
- Faça um push: `git push origin master`

▣ Parte 2

- Vá para o repositório “clone 2”
- Modifique o arquivo devtitans2.txt, adicione e dê commit
- Faça um push: `git push origin master`

Laboratório!

- ▣ Instale o PuTTY (cliente SSH):
 - ▣ bit.ly/putty-baixar
- ▣ Na janela "PuTTY Configuration"
 - ▣ Host Name: tardis.icomp.ufam.edu.br
 - ▣ Port: 8080
 - ▣ Clique em "Open" -> "Yes"
- ▣ Será pedido o login e a senha:
 - ▣ Login: iniciais do seu nome (Ex: yfa)
 - ▣ Senha: Titan@<iniciais>123 (Ex: Titan@yfa123)
- ▣ Por fim, entre no Moodle:
 - ▣ <https://bit.ly/devtitans-moodle>

