Lessons on Linux, Server and Git

Jauntyliu @ SCGY-Tech

October 27, 2018

What is Linux?

Linux:

- an operating system kernel written by Linus Torvalds in 1991
- components of the most famous open-source operating system
- highly scalable, customizable and stable
- distributes along with many free software
- full source code available at kernel.org

Linux Distributions

Linux itself is just a kernel. Without various applications, the OS wouldn't be usable.
A combination of Linux (the kernel) and its applications formed a **distribution**.

- Ubuntu
- Debian
- Fedora
- RHEL & CentOS
- Arch Linux
- Gentoo & Linux From Scratch
- Slax, Kali, Deepin, Tiny Core Linux...

OSS License

In open-source software, the freedom of users is guaranteed by software licenses.

- GNU General Public License
- GNU Lesser General Public License
- Apache & MIT & BSD License
- ...

Basic Environment Setup

Today, I'll share my Raspberry Pi 3B+ as a Linux host with you to enjoy. To connect:

1. Download PuTTY from my homepage
2. Extract and run PUTTY.EXE, fill the Host Name field with **192.168.2.141**
3. Type **tempuser** as "*Login as:*" appears
4. Type **233** as "*Password:*" appears.
   Notice: the password you type **will not be shown** on screen as Unix tradition

Basic Shell Commands

Format: **path_to_executable** *argument0* **argument1** $\cdots$
Example:

- cp a.txt b.txt
  Here, **cp** is the shorthand of **/usr/bin/cp**, and **a.txt** is the first argument, **b.txt** the second.
- ls
  A program can run without any arguments given.

Unix Paths

The Unix paths consists of two types:

1. Absolute Path : Always start with /
   - (a folder)**/usr/bin**
   - (a file)**/usr/bin/ls**

   Unlike disk C: and D: in Windows, there is only a single **Root** (/) in Linux, and real drives are mounted as folders of the root filesystem.
2. Relative Path: A **part** of the path, only useful when given the **current working directory**.
   Suppose you're in **/usr** (a folder)
   - **bin/ls** - the same as /usr/bin/ls
   - **.** - the dots means "*current directory*", hence "**/usr/**" itself
   - **..** - the double dots means "*parent directory*", hence "**/**"
   - **../home/../dev/../usr** - also means **/usr**

Basic Shell Commands

- ls - **list directories**
- cd - **change directory**
- pwd - **print working directory**
- mv - **move file**
- cp - **copy file**
- rm - **remove file**
- mkdir - **create a directory**
- man - **manual pages**
- apropos - **looking for relative commands and programs**

Mission 1: Create a directory called "**XcxSaikou**" in your home directory and copy **/etc/apt/sources.list** into the folder you create.

Basic Shell Commands

How to get help?

1. **the_commands_you_don't_understand –help**
2. **man the_commands_you_don't_understand**
3. **apropos the_keyword_about_what_you_want**
4. Google & Stack Overflow
5. Ask in SCGY-Tech or USTCLUG

Basic Shell Commands

One possible solution:

1. mkdir /XcxSaikou
2. cp /etc/apt/source.list /XcxSaikou
3. cd /XcxSaikou
4. ls -l

What's under /?

Mission 2: Find what's under "/"?

What's under /?

- **/home** - User home directories
  In Unix, the directory **/home/your_account** is the most decent place for you to store your personal data.
- **/usr** - Executables, libraries, and shared resources that are not system critical
- **/etc** - System-wide configuration files and system databases
- **/dev** - Devices, such as hard disks, ttys and displays.
- **/var** - Log files, print jobs, mails and temporaries
- **/lib** - Shared libraries, kernel module or device drivers.
- **/bin** - Binaries that should be available even when **/usr** haven't been mounted
- ...

Basic File Editing

- User-friendly: **nano a.txt**
- Experienced: **vim a.txt**
- Obsolete: **ed a.txt**

Mission 3: Create a text file called "**Comments_On_Fruits.txt**" and write some words in it. You may use any of them if you like.

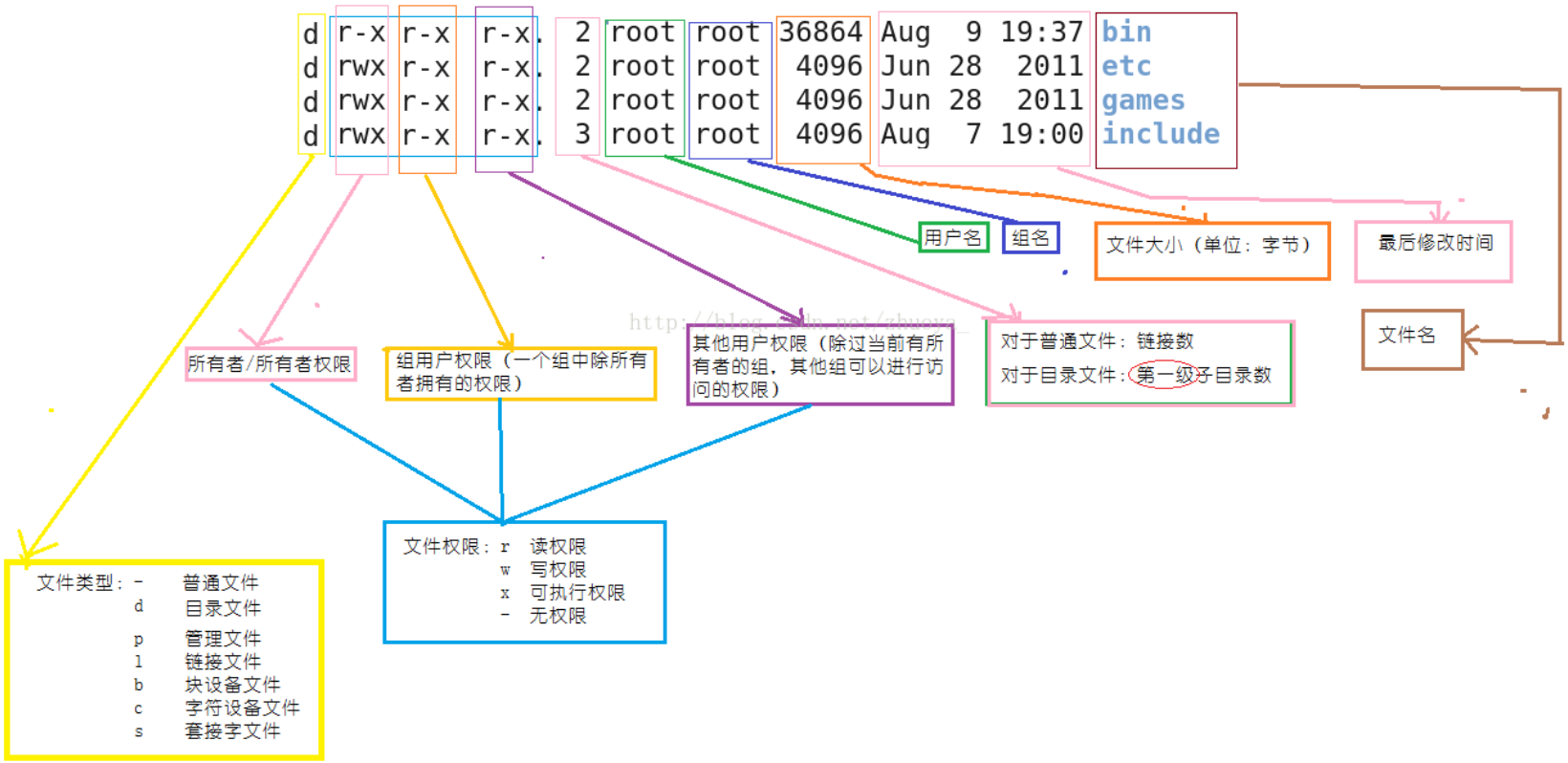
Users and Permissions

Users are entities who operate on this system.

- Every file and folder has an owner
- Users can only Read/Write/eXecute when they have the permission
- Use **ls -l** to see the permissions:

```
-rw-r--r--   1  libreliu  sudo   38 Oct 15 23:29   flag.txt
drwxr-xr-x   5  libreliu  sudo   4096 Oct 18 20:37  writeup
```

在Linux中执行命令ls -l（以列表的形式显示文件）后，文件中各个信息代表的含义图解



Users and permissions

Edit file permissions:

- **chmod three_octal_digits filename** - change permission bits
- **chown owner_name filename** - change file and directory owner
  Notice: changing the directory along **WON'T** affect its subdirectories and files.

Edit users and their passwords:

- **useradd** - add user to the system
- **userdel** - delete user
- **usermod** - modify user
- **passwd** - change user password
- **groupadd** - user groups related operations

Mount and umount in Unix

- A storage device must be mounted first before it's available for use. **At the system's perspective, it needs to know which folder the files of the disk should be in.**
- **lsblk** - list block devices, for you to see if your storage device is recognized by the system

```
NAME   MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda      8:0    0 298.1G  0 disk
|-sda1   8:1    0   487M  0 part /boot
|-sda2   8:2    0     1K  0 part
|-sda5   8:5    0  18.6G  0 part /
|-sda6   8:6    0  74.5G  0 part /home
`-sda7   8:7    0   2.8G  0 part [SWAP]
```

- **mount [-fnrsvw] [-t fstype] [-o options] device dir** - mount a device to a exist empty dir

Mount and umount in Unix

- **umount directory** or **umount device** - detaches the mentioned file system(s) from the file hierarchy

Programming under Linux

Programming under Linux boasts great efficiency

1. **sudo apt install build-essential** - Install gcc, make, g++ and so on.
2. **nano hello.c** - Write C Programs
3. **gcc hello.c -o hello && ./hello** - Compile and run the program
   && is just a way to execute two shell commands in a row, just like **if(c=1 && b=1 && a!=3)** in C

When in doubt

- **man 3 printf** - Check for manual pages about library function (section 3) **printf**

Installing software

For simple software like Browsers and Games:

1. Use Package Manager, search for it and then install the package (Described later)
2. Build from source: **./configure && make && sudo make install**
3. When bumping into dependency problems, try installing the missing software
4. When you've tried everything, discuss with others or open an issue to the project (forum) directly

For complicated, highly configurable software:

1. Looking for things like "**How to install and configure xxx**" on the Internet
2. Follow the instructions

Homework: Try building **nginx** web server from source.

Package Manager

Package manager helps you install software easily, without having to solve dependencies on your own, as well as upgrading the software by hand

- apt for Debian/Ubuntu
- yum & rpm for Fedora/OpenSUSE/CentOS/RHEL
- pacman for Arch Linux

They share some similar traits:

- Configurations for Software Sources - Integrated database for software packages
  See *USTCLUG Mirrors*
- **apt install software_name**
- **pacman -S software_name**
- See manpages for more details

SSH Operations

SSH stands for *Secure Shell*, a software suite and a protocol designed for delivering remote shell, file copying and port-forwarding.

- **ssh tempuser@192.168.4.233** - initiate a connection to remote machine, with username tempuser.
  When username was not specified, the ssh client will attempts to connect with local username that's using.
- **scp -r tempuser@192.168.4.233:/home/tempuser /home/tempuser/archive_at_remote** - copy files from remote machine by using *Secure Copy* command

Mission 4: Try copying files **flag.txt** from my laptop by using the account I given.

Git and Github

TBC, because I think I won't made it there in the first lecture.