



Linux System Programming

Part 1 - Linux Basics

IBA Bulgaria
2021



What is System Programming?

- The art of writing **system software**.
- **System software** lives at a low level, interfacing directly with the kernel and core system libraries.
- Assists general use application programs.
- **System programmers** must have a strong awareness of the hardware and operating system on which they are working.
- In Linux usually the system utilities are in `'/sbin'` and `'/usr/sbin'`.

Application Software

Text Processor, Internet Browser, etc.

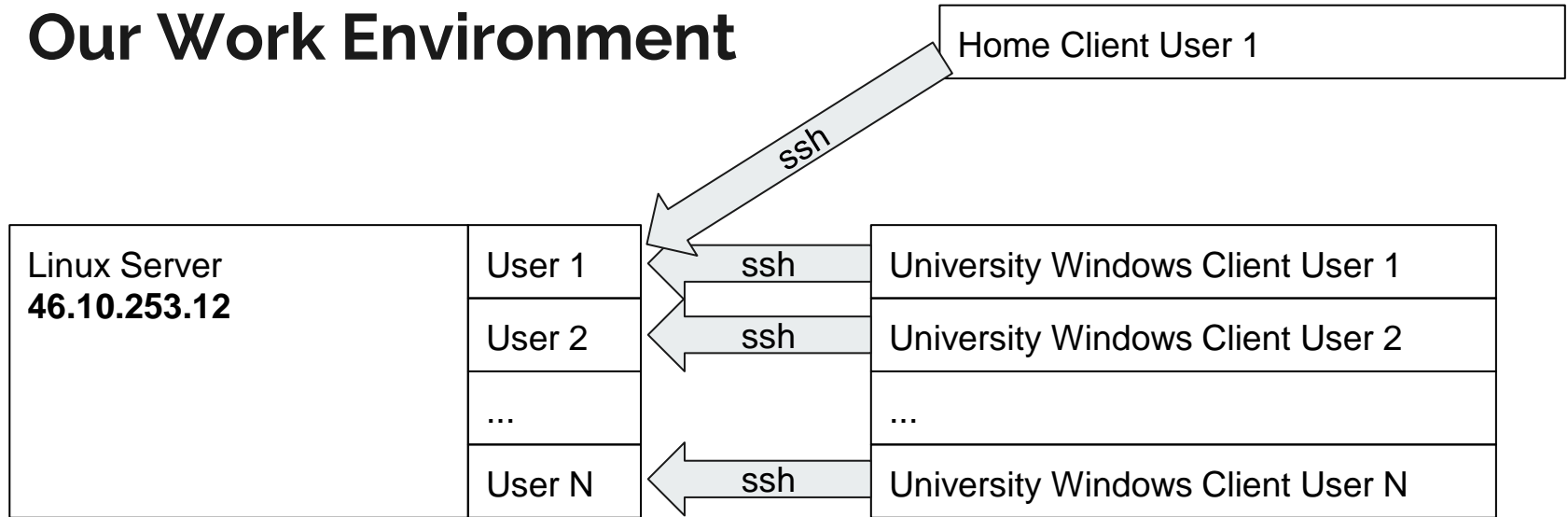
System Software

OS, Utilities, Drivers

Hardware

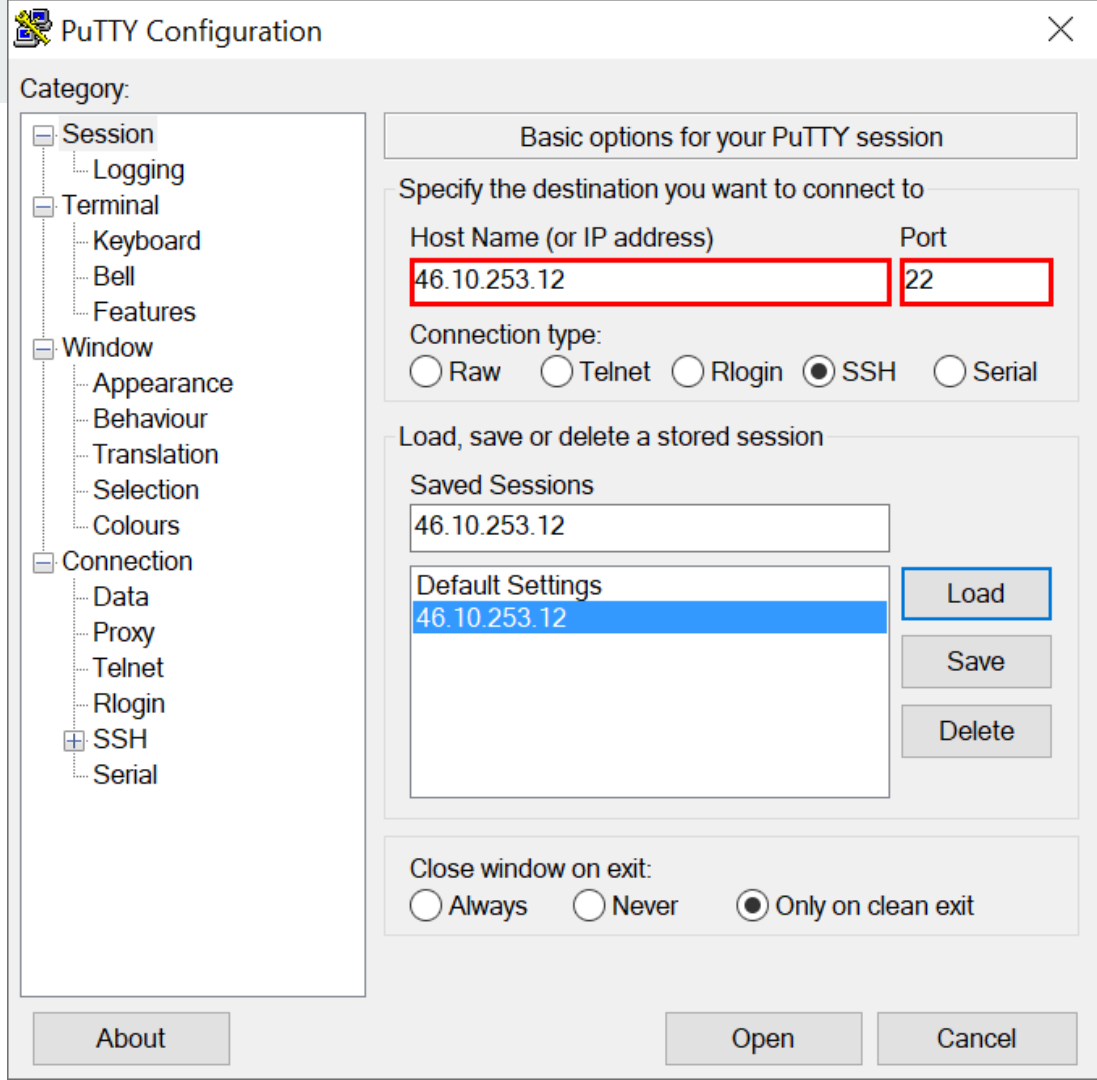
CPU, Memory, Disks, Keyboard, etc.

Our Work Environment



Remote Connection

- telnet (**T**ele**t**ype **N**etwork)
- ssh (**S**ecure **S**hell)
- PuTTY - has no official meaning.



How to get Help in Linux?

- man - an interface to the on-line reference manuals.
- man [option(s)] keyword(s)
- man man
- Most important keys :PgUp, PgDn, q

Of course you could always google it.

```
MAN(1)                                Manual pager utils                                MAN(1)

NAME
    man - an interface to the on-line reference manuals

SYNOPSIS
    man [-C file] [-d] [-D] [--warnings[=warnings]] [-R encoding] [-L locale] [-m system[,...]] [-M
    path] [-S list] [-e extension] [-i|-I] [--regex|--wildcard] [--names-only] [-a] [-u] [--no-sub
    pages] [-P pager] [-r prompt] [-7] [-E encoding] [--no-hyphenation] [--no-justification] [-p
    string] [-t] [-T[device]] [-H[browser]] [-X[dpi]] [-Z] [[section] page[.section] ...] ...
    man -k [apropos options] regexp ...
    man -K [-w|-W] [-S list] [-i|-I] [--regex] [section] term ...
    man -f [whatis options] page ...
    man -l [-C file] [-d] [-D] [--warnings[=warnings]] [-R encoding] [-L locale] [-P pager] [-r
    prompt] [-7] [-E encoding] [-p string] [-t] [-T[device]] [-H[browser]] [-X[dpi]] [-Z] file ...
    man -w|-W [-C file] [-d] [-D] page ...
    man -c [-C file] [-d] [-D] page ...
    man [-?V]

DESCRIPTION
    man is the system's manual pager. Each page argument given to man is normally the name of a
    program, utility or function. The manual page associated with each of these arguments is then
    found and displayed. A section, if provided, will direct man to look only in that section of
    Manual page man(1) line 1 (press h for help or q to quit)
```



Files and Directories in Linux

- **pwd** - print the name of the current working directory.
- **ls** - list directory contents.
- **mkdir** - make directories.
- **cd** - change the shell working directory.
- **rmdir** - remove directory, this command will only work if the folders are empty.
- **cp** - copy files and directories.
- **mv** - move (rename) files.
- **rm** - remove files or directories.
- **ln** - make links between files.
- **cat** - concatenate files and print on the standard output.


Transferring Files

- The File Transfer Protocol (FTP) is a standard network protocol used for the transfer of computer files between a client and server on a computer network.
- You can copy files into your '`~/ftp/files`' directory.
- You can download those files with any browser - '`ftp://46.10.253.12`' and login with your Linux credentials.
- You can manage your '`~/ftp/files`' directory with any **ftp client** (like FileZilla).

```
student@ubuntu:~$ pwd
/home/student
student@ubuntu:~$ ls
ftp
student@ubuntu:~$ ls ftp/
files
student@ubuntu:~$
```

← → ↻ ⓘ Not secure | ftp://46.10.253.12

Index of /

Name	Size	Date modified
 LSP.pdf	2.7 MB	3/16/20, 10:28:00 AM



Example

Files and directories management



Processes in Linux

- **Processes** are object code in execution: active, alive, running programs.
- Processes consist of **data**, **resources**, **state**, and a **virtualized computer**.
- Each process is represented by a **unique identifier**, the process ID (**pid**).
- The process that the kernel "runs" when there are no other runnable processes is the **idle process** (pid=0).
- The process that spawns a new process is known as the **parent**; the new process is known as the **child**.
- Each process is owned by a **user** and a **group**.

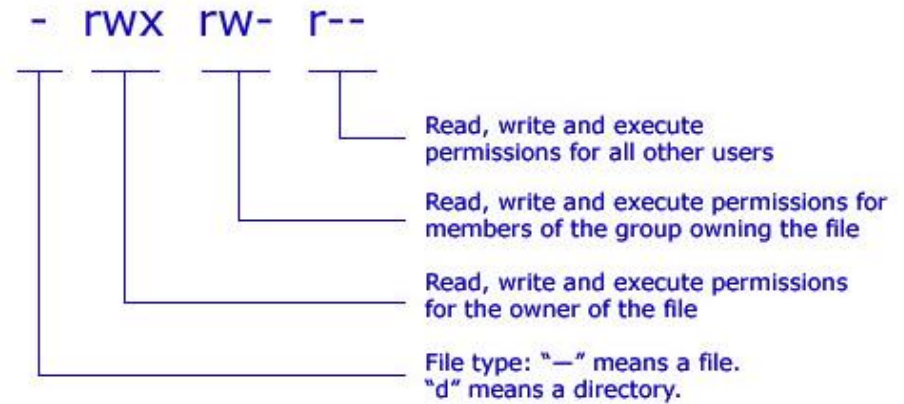


Users and Groups

- Authorization in Linux is provided by **users** and **groups**.
- Each user is associated with a unique positive integer called the user ID (**uid**).
- The users are often referred by **usernames**, not numerical values.
- Each user may belong to one or more groups, including a **primary** or **login group**.

File Permissions

- The traditional system defines three modes to use a file - **(r)**ead, **(w)**rite or e**(x)**ecute.
- There are three levels of access rights for the file operations - **owner**, **group** and **public**.
- Set-user Identification (**SUID**) - identified with **(s)** instead of (x) at owner level.
- When a command has SUID bit set, its **effective UID** becomes that of the owner of the file, rather than of the user who is running it.
- `chmod 4555 [path_to_file]`



```
sysadmin@ubuntu:~$ ls /usr/bin -al
total 103716
drwxr-xr-x  2 root  root   32768 Jun  1 15:29 .
drwxr-xr-x 10 root  root   4096 May  8 15:48 ..
-rwxr-xr-x  1 root  root   54772 Oct  4 2017 [
-rwxr-xr-x  1 root  root    96 Oct  4 2017 2to3-3.6
-rwxr-xr-x  1 root  root  22008 Sep 15 2017 aa-enabled
-rwxr-xr-x  1 root  root  22104 Sep 15 2017 aa-exec
-rwxr-xr-x  1 root  root   9984 Apr 28 2017 acpi_listen
-rwxr-xr-x  1 root  root   3452 Jul 20 2017 activate-global-python-argcomp
lete3
-rwxr-xr-x  1 root  root    6473 Oct 17 2017 add-apt-repository
-rwxr-xr-x  1 root  root   21916 Aug 14 2017 addpart
lrwxrwxrwx  1 root  root    24 Oct  1 2017 addr2line -> i686-linux-gnu-ad
dr2line
-rwxr-xr-x  1 root  root   2555 Oct 24 2017 apport-bug
-rwxr-xr-x  1 root  root  13348 May 11 01:27 apport-cli
lrwxrwxrwx  1 root  root    10 May 11 01:27 apport-collect -> apport-bug
-rwxr-xr-x  1 root  root   1849 May 11 01:27 apport-unpack
lrwxrwxrwx  1 root  root     6 Dec 13 2016 apropos -> whatis
-rwxr-xr-x  1 root  root   13732 Oct 26 2017 apt
```

File Permissions

- Set-group identification (**SGID**) - identified with **(s)** instead of (x) at group level.
- When a command with **SGID** is run, it runs as if it were a member of the same group in which the file is a member.
- `chmod 2555 [path_to_file]`
- When **SGID** permission is set on a directory, files created in the directory belong to the group of which the directory is a member.
- `chmod g+s [path_to_directory]`
- **Sticky Bit** - identified with **(t)** instead of (x) at public level and is primarily used on shared directories. Users can create files, read and execute files owned by other users, but are not allowed to remove files owned by other users.
- `chmod +t [path_to_directory]`

ls /usr/bin -al

```
sysadmin@ubuntu:~$ ls /usr/bin -al
total 103716
drwxr-xr-x  2 root  root   32768 Jun  1 15:29 .
drwxr-xr-x 10 root  root   4096 May  8 15:48 ..
-rwxr-xr-x  1 root  root   54772 Oct  4 2017 [
-rwxr-xr-x  1 root  root    96 Oct  4 2017 2to3-3.6
-rwxr-xr-x  1 root  root  22008 Sep 15 2017 aa-enabled
-rwxr-xr-x  1 root  root  22104 Sep 15 2017 aa-exec
-rwxr-xr-x  1 root  root   9984 Apr 28 2017 acpi_listen
-rwxr-xr-x  1 root  root   3452 Jul 20 2017 activate-global-python-argcomp
lete3
-rwxr-xr-x  1 root  root    6473 Oct 17 2017 add-apt-repository
-rwxr-xr-x  1 root  root   21916 Aug 14 2017 addpart
lrwxrwxrwx  1 root  root    24 Oct  1 2017 addr2line -> i686-linux-gnu-ad
dr2line
-rwxr-xr-x  1 root  root    2555 Oct 24 2017 apport-bug
-rwxr-xr-x  1 root  root   13348 May 11 01:27 apport-cli
lrwxrwxrwx  1 root  root    10 May 11 01:27 apport-collect -> apport-bug
-rwxr-xr-x  1 root  root   1849 May 11 01:27 apport-unpack
lrwxrwxrwx  1 root  root    6 Dec 13 2016 apropos -> whatis
-rwxr-xr-x  1 root  root   13732 Oct 26 2017 apt
```



Managing Users and Groups

- **id** - print real and effective user and group IDs.
- **chmod** - change file mode bits.
- **umask** - set file mode creation mask.
- **chown** - change file owner and group.
- **chgrp** - change group ownership.
- **passwd** - change user password.



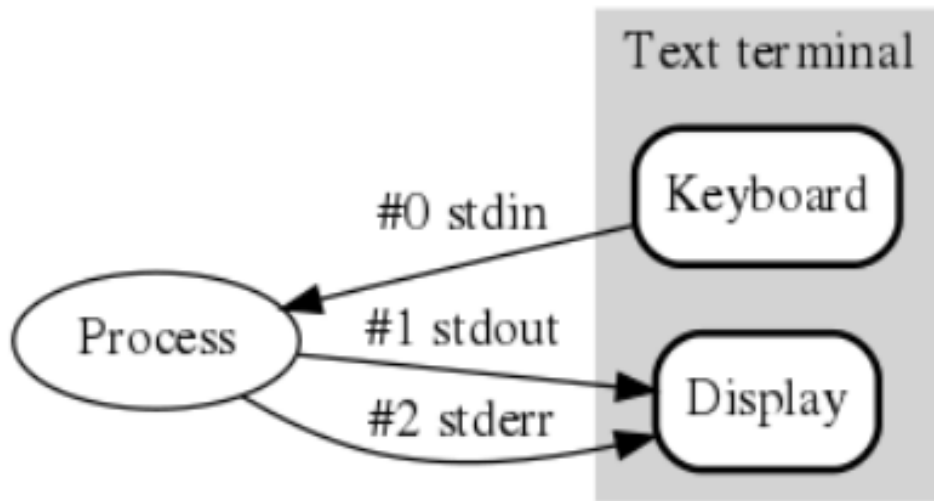
Example

Manage users and groups.

Standard Streams

Every program we run on the command line automatically has three data streams connected to it.

- **STDIN** (0) - Standard input (data fed into the program)
- **STDOUT** (1) - Standard output (data printed by the program, defaults to the terminal)
- **STDERR** (2) - Standard error (for error messages, also defaults to the terminal)





Redirection and Piping

- The greater than operator (`>`) redirects programs output (or whatever it sends to STDOUT) to be saved in a file instead of printed to the screen.
- If we save into a file which already exists, it's contents will be cleared, then the new output saved to it.
- The double greater than operator (`>>`) redirects the output and appends it to a file.
- The piping operator (`|`) feeds the output from the program on the left as input to the program on the right.

```
user@bash: ls
barry.txt bob example.png firstfile foo1 video.mpeg
user@bash: ls > myoutput
user@bash: ls
barry.txt bob example.png firstfile foo1 myoutput video.mpeg
user@bash: cat myoutput
barry.txt
bob
example.png
firstfile
foo1
myoutput
video.mpeg
```

```
user@bash: ls
barry.txt bob example.png firstfile foo1 myoutput video.mpeg
user@bash: ls | head -3
barry.txt
bob
example.png
```

```
user@bash: ls | head -3 | tail -1
example.png
user@bash:
```




Nano text editor

- **nano** - Nano's ANOther editor, an enhanced free Pico clone.
- select - drag mouse.
- paste - right mouse button.
- <https://www.howtogeek.com/howto/42980/the-beginners-guide-to-nano-the-linux-command-line-text-editor/>
- <http://www.codexpedia.com/text-editor/nano-text-editor-command-cheatsheet/>

File Control in nano

nano index.php Open or create the file "index.php" with nano on command line.

Ctrl-o Y Enter Save changes.

Ctrl-r Alt-f Open a new file with a new buffer within nano.

Alt-> Switch to the next file buffer in nano.

Alt-< Switch to the previous file buffer in nano.

Ctrl-x Quit nano.

Navigating through file contents in nano

Ctrl-a Move to the beginning of the current line.

Ctrl-e Move to the end of the current line.

Ctrl-v Move down one page.

Ctrl-y Move up one page.

Alt- Go to the beginning of the file.

Alt-/ Go to the end of the file.

Alt-g Go to a target line number.

Alt-] Jump to matching open/close symbol.

Alt-a Alt-} Select a block and indent the block.

Alt-a Alt-{ Select a block and outden the block.

Copy and Paste in nano

Alt-a To select a block for copy or cut operation, do Alt-a again to unselect.

Alt-a Alt-^ Copy a highlighted block to the clipboard.

Alt-a Ctrl-k Cut a highlighted block to the clipboard.

Ctrl-k Cut from the current cursor position to the end of the current line.

Ctrl-u Paste the contents from the clipboard at the current cursor position.

Search and Replace in nano

Ctrl-w Search for a target string.

Alt-w Repeat the last search.

Alt-r Search and replace.

Exercise



1. Establish a remote connection (with PuTTY) to our server (**46.10.253.12**), your initial password is '**password123**'.
2. Go to '**ftp/files**' (you can use the '**cd**' command) and copy there the file '**/home/students/ 01_Linux_Basics.pdf**' (you can use the '**cp**' command).
3. Open your ftp directory in a Windows browser (**ftp://46.10.253.12**) and download the '**01_Linux_Basics.pdf**' and open it to read this lecture.

In your Linux environment:

1. Change your password.
2. In your **home** ('~/') directory create a new one for the exercises (like 'exercises') and a sub-directory in it for this lecture ('day01').
3. Go to the last folder ('day01') and create a file ('listing.txt') there, which contains the list of the files in '**/sbin**' folder.
4. Open the listing file with **nano** and add your names at the beginning.
5. In your home directory create a **link** to the '**/home/students/**' directory.