Linux Shell Tutorial

2024 3 9

李云广

Linux Shell

√ Brief Intro: all you need to know about starting using a CLI

√ Basic but useful command line tools

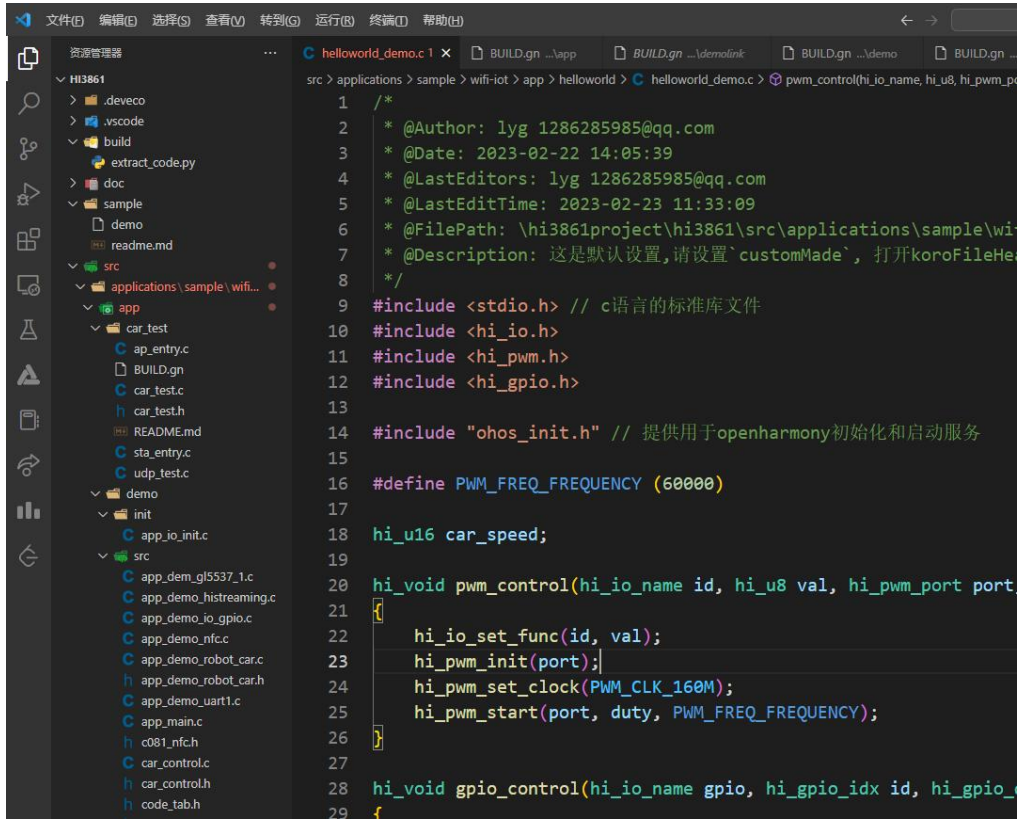
√ How to write a bash scripts and what can those scripts do?

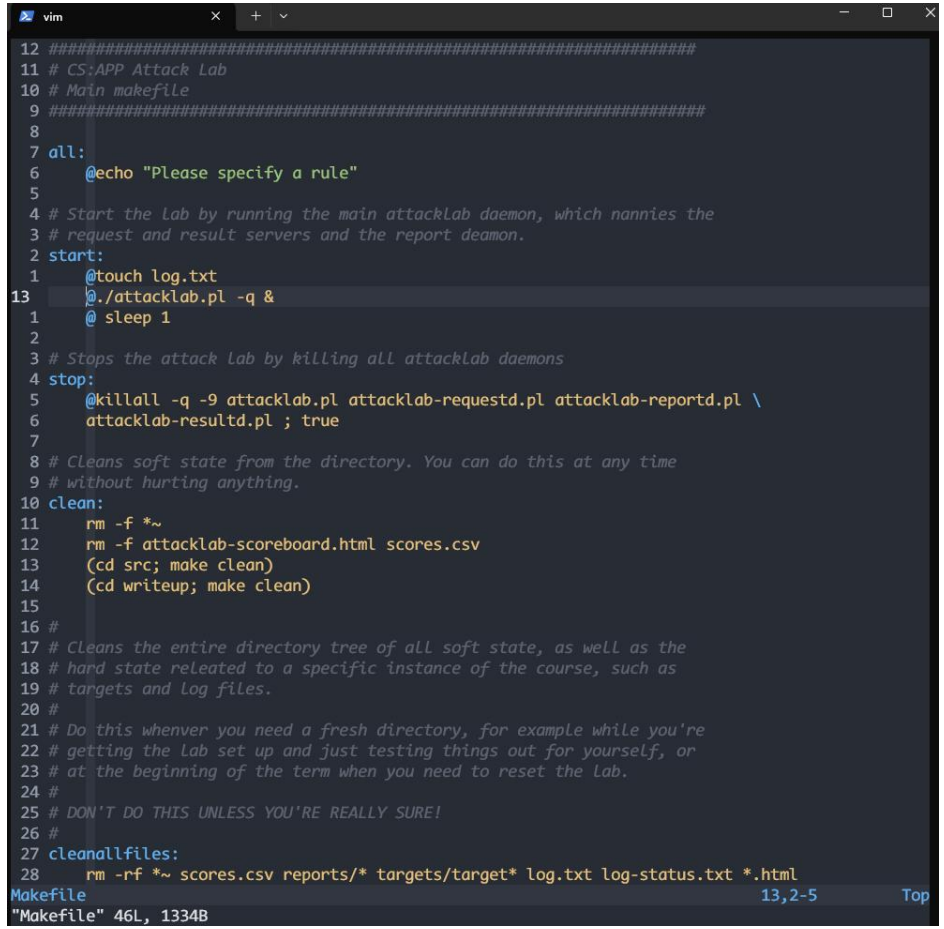
Target

1. Using shell to get the work done (doing experiment, coding, etc.) efficiently

2. Using bash script

GUI vs CLI: which is better?





GUI

- Graph (e.g., code analysis tools)
- More intuitive user-interface, especially in complex software

CLI

- Data analysis
- Home-brew app
- When connecting to a server

Shell: The system user-interface in CLI

Just like the Desktop in GUI world (from user's view)

Capability:

- Launch app
- execute command
- manage foreground/background tasks

Basic Setup

Terminal (emulator): emulate a (texted-based) terminal inside the GUI environment.

SSH to server ssh 2183311128–ics@igw.dfshan.net –p2291

Running sshd: daemon of SSH server

- **Strong password** or use **ssh key** to login
- • Keyboard shortcuts
  - ctrl + r (to find history), tab (to autofill)
  - ctrl + c (to kill SIGINT)

Install Software in CLI

Package manager: apt (ubuntu, Debian), brew (macOS), dnf (fedora)

- Search (e.g. apt search)
- https://command-not-found.com/

Build from source (no suitable version, or need to modify their code)

- README/INSTALL doc
- configure and make install

Basic Tools (Commands)

Directories： pwd, cd

File: touch, cp, mv, rm, cat, less, mkdir

Simple functions: sort, wc, echo

Others： grep, chmod

Code Editor: vim

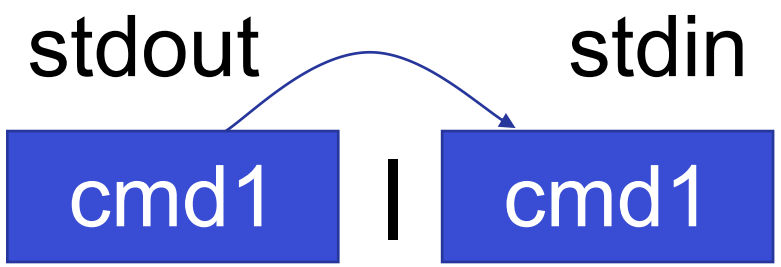
Keep the connection: tmux, screen, etc.

How to use?

- –help, ––help
- man [command]
- https://command-not-found.com/
- TLDR https://tldr.sh/

Communication: Pipe & Redirect

- **A lot of CLI tools, communication is required to do complex jobs**
- **Pipe: | use the stdout of previous command as the stdin of the next**



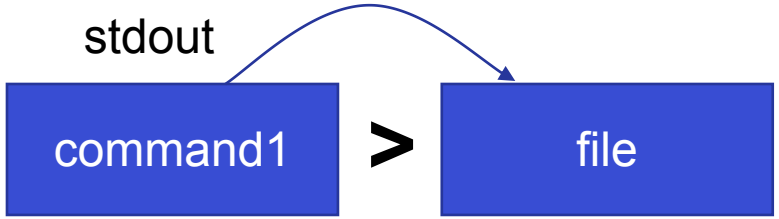

```
ygli@dc2:~/lab1$ ls | grep "sh"
lab1.sh
lab2.sh
rum_dummy.sh
run_exp.sh
test.sh
ygli@dc2:~/lab1$
```

ls | grep “sh”

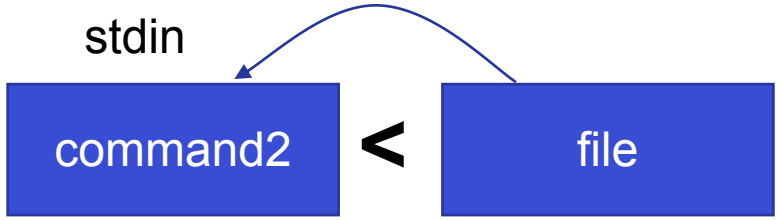
Communication: Pipe & Redirect

- **A lot of CLI tools, communication is required to do complex jobs**

- **Redirect: > & <, stdout to file or file to stdin (normally)**



ls > ls_out



grep sh < ls_out

```
ygli@dc2:~/lab1$ ls > ls_out
ygli@dc2:~/lab1$ grep sh < ls_out
lab1.sh
lab2.sh
rum_dummy.sh
run_exp.sh
test.sh
ygli@dc2:~/lab1$
```

Communication: Pipe & Redirect

- **A lot of CLI tools, communication is required to do complex jobs**

- **Redirect: > & <, stdout to file or file to stdin**

```
 1 #include <iostream>
 2 #include <stdio.h>
 3 using namespace std;
 4
 5 int main(){
 6     printf("stdout\n");
 7     fprintf(stdout, "stdout\n");
 8     fprintf(stderr, "stderr\n");
 9     return 0;
10 }
11
~
~
~
test1.cpp            11,0-1          All
```

0 - stdin, the standard input stream.
1 - stdout, the standard output stream.
2 - stderr, the standard error stream.

```
ygli@dc2:~/lab1$ g++ test1.cpp -o test1
ygli@dc2:~/lab1$ ./test1
stdout
stdout
stderr
ygli@dc2:~/lab1$ ./test1 > test_out
stderr
ygli@dc2:~/lab1$ ./test1 > test_out 2>test_err_out
ygli@dc2:~/lab1$ cat test_err_out
stderr
ygli@dc2:~/lab1$
```

Tar

Usage Scenario: archive files in 1 bundle (and compress them)

Flags

–c: create a tarball
–x: open a tarball
–z: zipped using gzip
–v: verbose mode [displays progress]
–f: specify file name

tar –cf name–of–archive.tar /path/to/dir/ compress directory
tar –cf name–of–archive.tar /path/to/filename compress file
tar –cf name–of–archive.tar dir1 dir2 dir3 compress multiple directories
tar –xf name–of–archive.tar open a tar file in current directory

Some useful tools

ag

awk

sed

shell script

touch lab1.sh

echo hello1
echo hello2
echo hello3

chmod +x lab1.sh
./lab1.sh

```
ygli@dc2:~/lab1$ cat lab1.sh
echo hello1
echo hello2
echo hello3

ygli@dc2:~/lab1$ ./lab1.sh
hello1
hello2
hello3
ygli@dc2:~/lab1$
```

shell script

- With local variables
- Passing in as an arguments

lab2.sh

```
hello1=$1
hello2=$2
hello3=$3

echo “$hello1”
echo “$hello2”
echo “$hello3”
```

$0 $1 $2 $3

```
ygli@dc2:~/lab1$ ./lab2.sh 1 2 3
1
2
3
ygli@dc2:~/lab1$ ./lab2.sh 4 5 6
4
5
6
```

Example

Running Experiments

```
#!/bin/bash
# 1-10 helloi.sh
mkdir hello
cd hello
for i in {1..10}; do
        touch hello$i.sh
        chmod +x hello$i.sh
        echo "echo hello$i" > hello$i.sh
done
                                4,8           All
```

```
#!/bin/bash
for i in {1..10}; do
        ./hello/hello$i.sh
done
~
~
~
~
```

The best way to learn it,
is to use it.

Happy Shell-ing!

Learn more

https://www.bilibili.com/video/BV1y44y1v7c3/

https://www.bilibili.com/video/BV14E411J7n2/