C_wshell

October 4, 2023

1 Programming with Linux

BY WROX

Normally when we are not working in notebook setting we need to compile the program

```
$ gcc -o hello hello.c
$ ./hello
```

```
Hello World
```

[ ]:
```
%%bash
gcc --version

# compile hello.c into hello binary then run it
gcc -o hello hello.c
./hello
```

```
gcc (Ubuntu 9.4.0-1ubuntu1~20.04.2) 9.4.0
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

Hello, world!
```

> If you forget the `-o` flag then compiled program would go to `a.out` meaning assembler output *this comes from early days of UNIX when people would play games as* ***`a.out`*** *to avoid being caught by system administrators*

1.1 Directory Structure

1.1.1 Applications

`/usr/bin` supplied by system for general use including program development

`/usr/local/bin` or `/opt` applications added by sys admins for a specific host computer or local network

[ ]:
```
%%bash
echo "--- /usr/bin ---"
ls /usr/bin | tail
echo "--- /usr/local/bin ---"
```

```
ls /usr/local/bin | tail
```

1.1.2 Header Files

Usually located within `/usr/include` for C `usr/include/sys` adn `usr/include/linux`

[ ]:
```
%%bash
echo "--- /usr/include ---"
ls /usr/include | tail
echo "--- /usr/include/sys ---"
ls /usr/include/sys | tail
echo "--- /usr/include/linux ---"
ls /usr/include/linux | head
```

[ ]:
```
%%bash
cd /usr/include
grep EXIT_ *.h | head
```

1.1.3 Libraries

Libraries are collections of precompiled functions that have been written to be reusable #### Stored In `/lib` and `/usr/lib`

- `.a` for traditional static libraries
- `.so` for shared libraries

[ ]:
```
%%bash
echo "--- /lib ---"
ls /lib | head
echo "--- /usr/lib ---"
ls /usr/lib | head
```

1.2 Static Libraries

`fred.c`

```
#include <stdio.h>
void fred(int arg) {
  printf("fred: you passed %d\n", arg);
}
```

`bill.c`

```
#include <stdio.h>
void bill(char *arg) {
  printf("bill: you passed %s\n", arg);
}
```

[ ]:
```
%%bash
gcc -c bill.c fred.c
```

```
ls *.o
```

> `c` flag prevents compiler from created a complete program which is needed because `main` function has not been defined

We should create a header file

`lib.h`

```
/*
This is lib.h. It declares the functions fred and bill for users
*/
void bill(char *);
void fred(int);
```

We can include this header file in the calling program

`program.c`

```
#include "lib.h"
int main() {
    bill("Hello World");
    exit(0);
}
```

[ ]:
```
%%bash
gcc -c program.c
gcc -o program program.o bill.o
./program
```

[ ]:
```
%%bash
date
```

```
Mon 02 Oct 2023 07:02:47 PM EDT
```

1.3 who command

Explanation of the `who` command

```
username     terminal being used         date and time that each user logged in
```

1.3.1 Options

`-H --heading` display headings
`-q --count` quick who just display name and number of users
`-b` Displays the time and date of the last reboot
`--help`

[ ]:
```
%%sh
who -H # lists the login names, terminal lines, and login times of the users␣
  ↪who are currently logged on the system
```

1.3.2 Display Calender

cal

[ ]:
```
%%sh
cal
```

```
   October 2023
Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31
```

`learn` not installed on this system `help` much better than learn

[ ]:
```
%%bash
help
```

```
GNU bash, version 5.0.17(1)-release (x86_64-pc-linux-gnu)
These shell commands are defined internally.  Type `help' to see this list.
Type `help name' to find out more about the function `name'.
Use `info bash' to find out more about the shell in general.
Use `man -k' or `info' to find out more about commands not in this list.

A star (*) next to a name means that the command is disabled.

 job_spec [&]                            history [-c] [-d offset] [n] or hist>
 (( expression ))                        if COMMANDS; then COMMANDS; [ elif C>
 . filename [arguments]                  jobs [-lnprs] [jobspec …] or jobs >
 :                                       kill [-s sigspec | -n signum | -sigs>
 [ arg… ]                              let arg [arg …]
 [[ expression ]]                        local [option] name[=value] …
 alias [-p] [name[=value] … ]          logout [n]
 bg [job_spec …]                       mapfile [-d delim] [-n count] [-O or>
 bind [-lpsvPSVX] [-m keymap] [-f file>  popd [-n] [+N | -N]
 break [n]                               printf [-v var] format [arguments]
 builtin [shell-builtin [arg …]]       pushd [-n] [+N | -N | dir]
 caller [expr]                           pwd [-LP]
 case WORD in [PATTERN [| PATTERN]…)>  read [-ers] [-a array] [-d delim] [->
 cd [-L|[-P [-e]] [-@]] [dir]            readarray [-d delim] [-n count] [-O >
 command [-pVv] command [arg …]        readonly [-aAf] [name[=value] …] o>
 compgen [-abcdefgjksuv] [-o option] [>  return [n]
 complete [-abcdefgjksuv] [-pr] [-DEI]>  select NAME [in WORDS … ;] do COMM>
 compopt [-o|+o option] [-DEI] [name .>  set [-abefhkmnptuvxBCHP] [-o option->
 continue [n]                            shift [n]
 coproc [NAME] command [redirections]    shopt [-pqsu] [-o] [optname …]
```

```
declare [-aAfFgilnrtux] [-p] [name[=v>  source filename [arguments]
dirs [-clpv] [+N] [-N]                   suspend [-f]
disown [-h] [-ar] [jobspec … | pid >  test [expr]
echo [-neE] [arg …]                    time [-p] pipeline
enable [-a] [-dnps] [-f filename] [na>  times
eval [arg …]                           trap [-lp] [[arg] signal_spec …]
exec [-cl] [-a name] [command [argume>  true
exit [n]                                type [-afptP] name [name …]
export [-fn] [name[=value] …] or ex>  typeset [-aAfFgilnrtux] [-p] name[=v>
false                                   ulimit [-SHabcdefiklmnpqrstuvxPT] [l>
fc [-e ename] [-lnr] [first] [last] o>  umask [-p] [-S] [mode]
fg [job_spec]                           unalias [-a] name [name …]
for NAME [in WORDS … ] ; do COMMAND>  unset [-f] [-v] [-n] [name …]
for (( exp1; exp2; exp3 )); do COMMAN>  until COMMANDS; do COMMANDS; done
function name { COMMANDS ; } or name >  variables - Names and meanings of so>
getopts optstring name [arg]            wait [-fn] [id …]
hash [-lr] [-p pathname] [-dt] [name >  while COMMANDS; do COMMANDS; done
help [-dms] [pattern …]                { COMMANDS ; }
```

[ ]:
```
%%sh
cal 9 1752 # see manual page below for explanation

# An unusual calendar is printed for September 1752. That is the month 11 days␣
↪were skipped to make up for lack of leap year adjustments. To see this␣
↪calendar, type: cal 9 1752
```

```
   September 1752
Su Mo Tu We Th Fr Sa
       1  2 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
```