【Linux Programming】Day3(2)

Tags	
Date	@May 15, 2022
Summary	

【Ch2】Shell Programming

2.3.2 Creating a Script

Using any text editor, we need to create a file containing the commands; creating a file called first that looks like this:

```
#!/bin/sh

# first
# This file looks through all the files in the current
# directory for the string POSIX, and then prints the names of
# those files to the standard output.

for file in *
do
  if grep -q POSIX $file
  then
    echo $file
  fi
done

exit 0
```

Comments start with a `#` and continue to the end of a line.

The first line, `#!/bin/sh`, is a special form of comment; the `#!` character tell the system that the argument that follows on the line is the program used to execute this file. In this case, `/bin/sh` is the default shell program.

Since the script is essentially treated as standard input to the shell, it can contain any Linux commands referenced by our PATH environment variable.

The exit command ensures that the script returns a sensible exit code .If we want to invoke one script from another script and check whether it succeeded, returning an appropriate exit code is very important.

2.3.4 Making a Script Executable

Now we have our script file, we can run it in two ways. The simpler way is to invoke the shell with the name of the script file as a parameter:

```
$ /bin/sh first
```

We can also use `chmod` command to make the file executable:

```
$ chmod +x first
```

We can then execute it using the command:

```
$ first
```

Once we are confident that our script is executing properly, we can move it to a more appropriate location.

```
# cp first /usr/local/bin
# chown root /usr/local/bin/first
# chgrp root /usr/local/bin/first
# chmod 0755 /usr/local/bin/first
```

Of alternatively to change mode:

```
#chmod u=rwx,go=rx /usr/local/bin/first
```