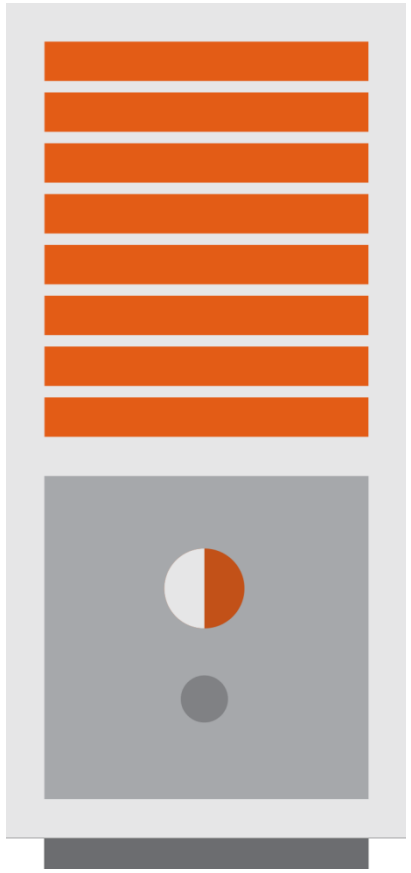
Build C Programs from Source

Module Overview

- Unpack files from archives
- Configure, Makefiles and make
- Compile C code
- Patching source code with patch and diff

Used in this Module



For this module we use predominately the **Raspberry Pi** which is a **Debian** based distribution. Package management differs to RHEL based systems however the basics of compiling from source are consistent.

We use **RHEL 7** to compile nmap from source.

Make and install programs from source

Archives and Compression Tools

Often source code will be packaged within an archive

LPI Objective 206.2 looks at backup solutions and is covered within a later module to this course

Source Packages

- The source for OSS packages can be downloaded from official repositories
- Ensure source repo is included in **/etc/apt/sources.list**

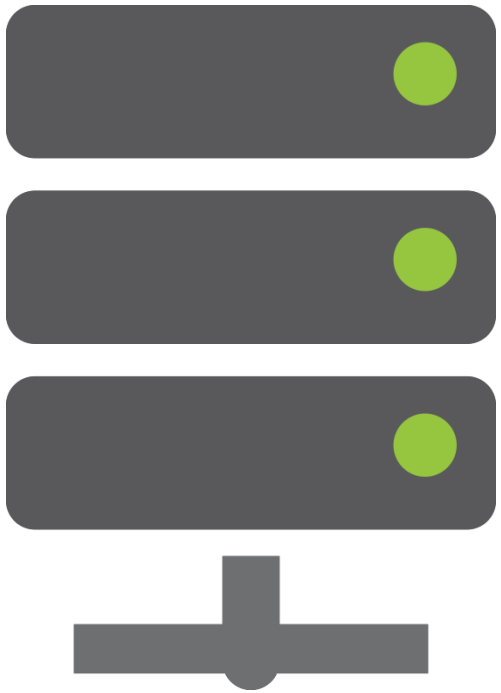
deb-src http://mirrordirector.raspbian.org/raspbian/ wheezy main contrib non-free rpi

```
$ sudo apt-get source nmap
$ ls /usr/src/nmap-6.00
```

Obtaining source packages

Here we install the source package for nmap, the port scanner

The package is added to **/usr/src/**<package name and version>



Demo: Configure and Use Source Repositories on Raspberry Pi

```
$ sudo yum groupinstall “Development Tools”   #RHEL
$ sudo apt-get install build-essential        #Debian
$ zypper install --type pattern devel_basis   #SUSE
```

Installing Software Compilation Tools

Depending on your platform the development tools we require can be installed with the following software groups

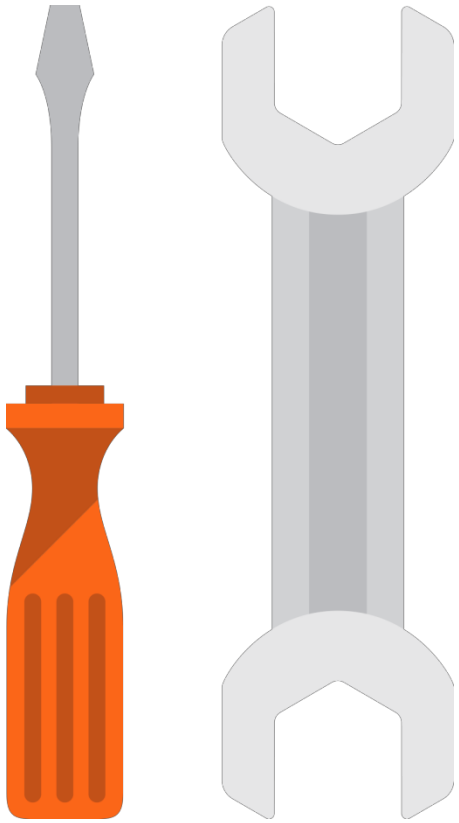
```
$ cd /tmp
$ wget http://nmap.org/dist/nmap-6.47.tar.bz2
$ tar -xjvf nmap-6.47.tar.bz2 && cd nmap-6.47
```

Obtain latest version of nmap

RHEL 7 ships with nmap version 6.40 the, currently, very latest version is close to that 6.47

If we want or need the very latest version we will need to download it from nmap and expand the archive

Compiling from Source



./configure

Configure script create the Makefile instruction set to compile for your system

--without-ndiff --without-zenmap --without-nping

options can be supplied to the configure script

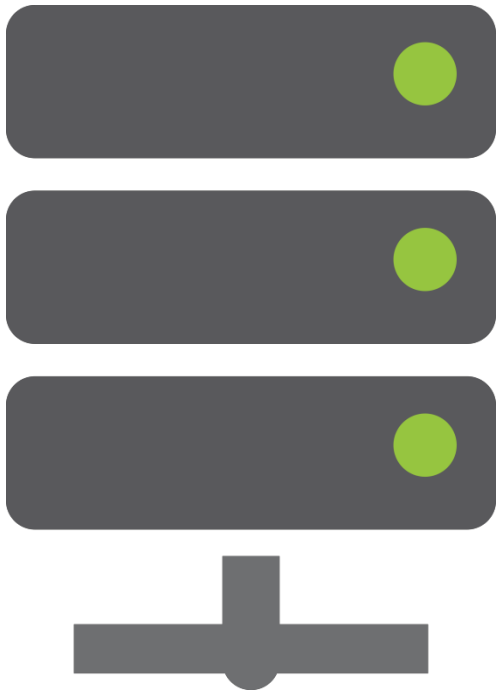
make

Looks for Makefile and instruction to compile

sudo make install

Checks Makefile for install actions

nmap --version

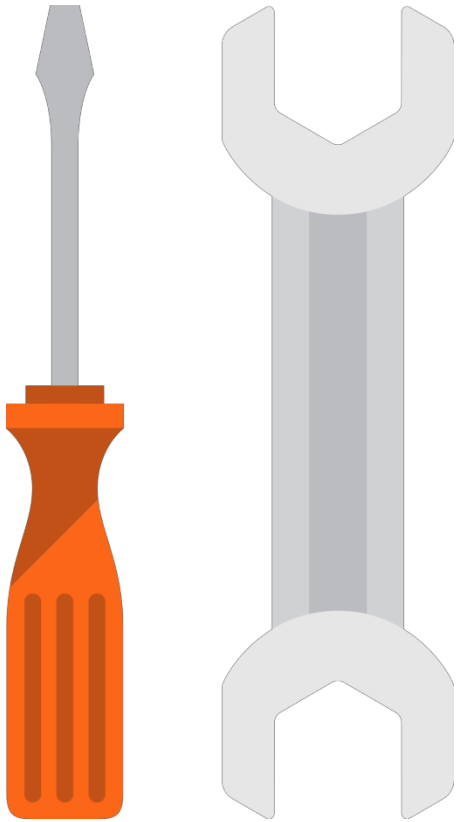


Demo: Compiling nmap 6.47 from Source

Compiling Your Own C Code

```
#include <stdio.h>
#include <string.h>
main () { /* yes this is poor code but we will patch later */
 char answer[3];
 printf("What is the gnu compiler called? ");
 fgets(answer,4, stdin);
 if ( strcmp (answer, "gcc") == 0 ) {
        printf("Correct\n");
        }
 else {
        printf("Sorry - gcc is correct\n");
        }
}
```

Using gcc

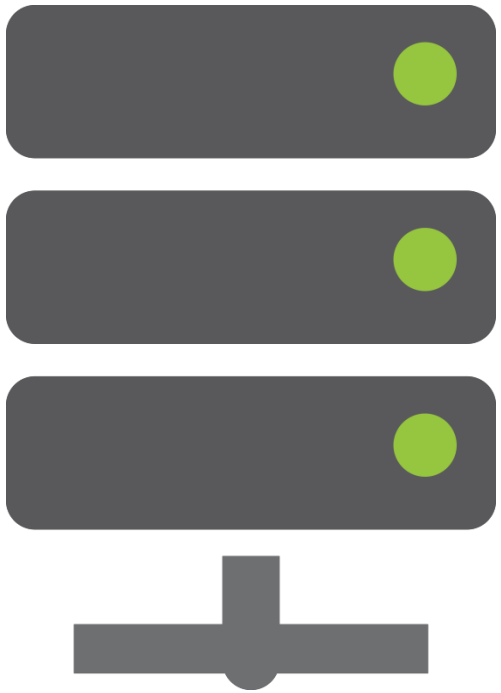


gcc question.c -o question

The code is compiled into the output binary

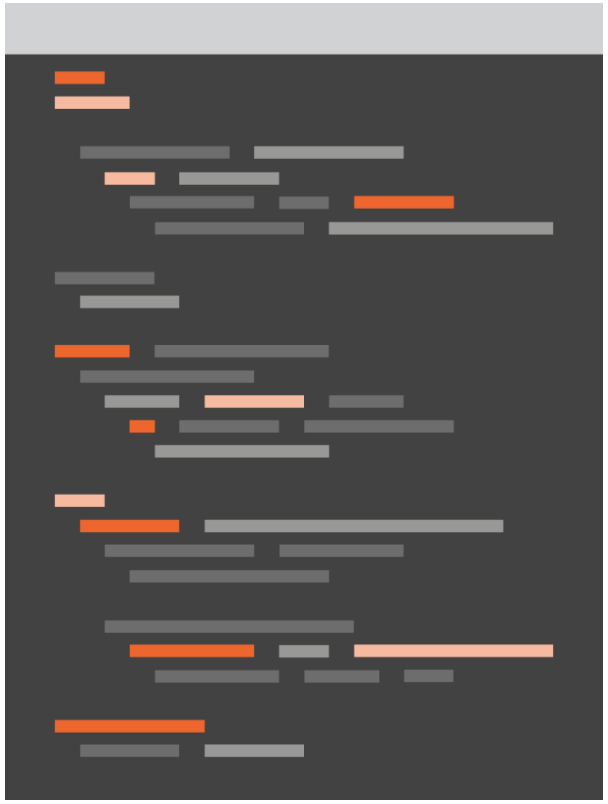
./question

It is compiled with the execute permission



Demo: Using gcc

Creating Patches



- Patches are used to update source code
- Copy the source to a new version file
- Edit the new source file
- Use diff to compare version and create patch file
- Distribute the patch

New Version

```
#include <stdio.h>
#include <string.h>
int main () {
char answer[3];
 printf("What is the gnu compiler called? ");
 fgets(answer,4, stdin);
 if ( strcmp (answer, "gcc") == 0 ) {
        printf("Correct\n");
        }
 else {
        printf("Sorry - gcc is correct\n");
        }
}
```

```
$ diff -u question.c questionv1.c > question.patch
```

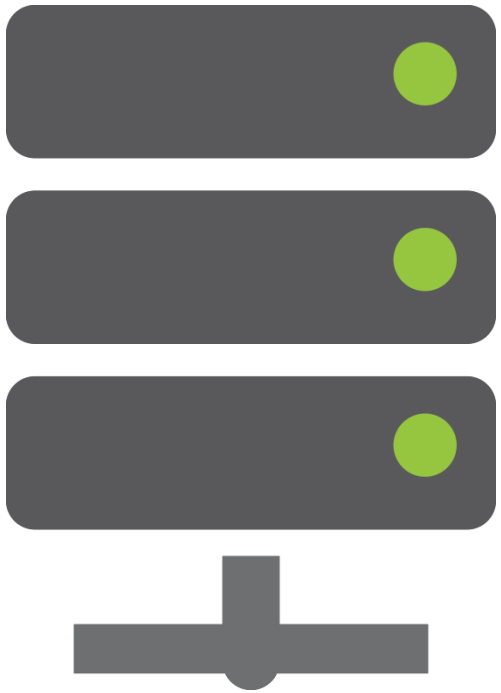
Create the patch file

The resulting patch is the difference between the original source and the new source and can be used to patch the original source version

```
$ cd /usr/src/c/question
$ mv /tmp/question.patch .
$ patch < question.patch
```

Patching the original source file

Copy the patch file to the correct directory and run the patch command reading in the patch



Demo: Using diff and patch

Summary

- Downloaded source deb packages
- Downloaded source code
- Unpacked and viewed the source
- Used configure scripts to create Makefiles for make to use
- Used diff and patch to patch source code

Next Up : Backing up data, LPI objective 206.2