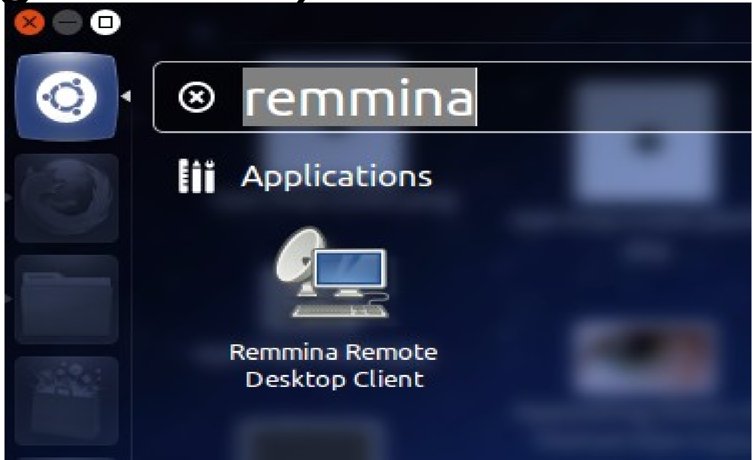
Raspbian (Linux)

VNC
Linux History
Overview
Shell
Commands
File system

A new way to connect the Pi: VNC

http://quaintproject.wordpress.com/2013/03/24/establish-a-vnc-connection-to-your-raspberry-pi-from-a-linux-pc

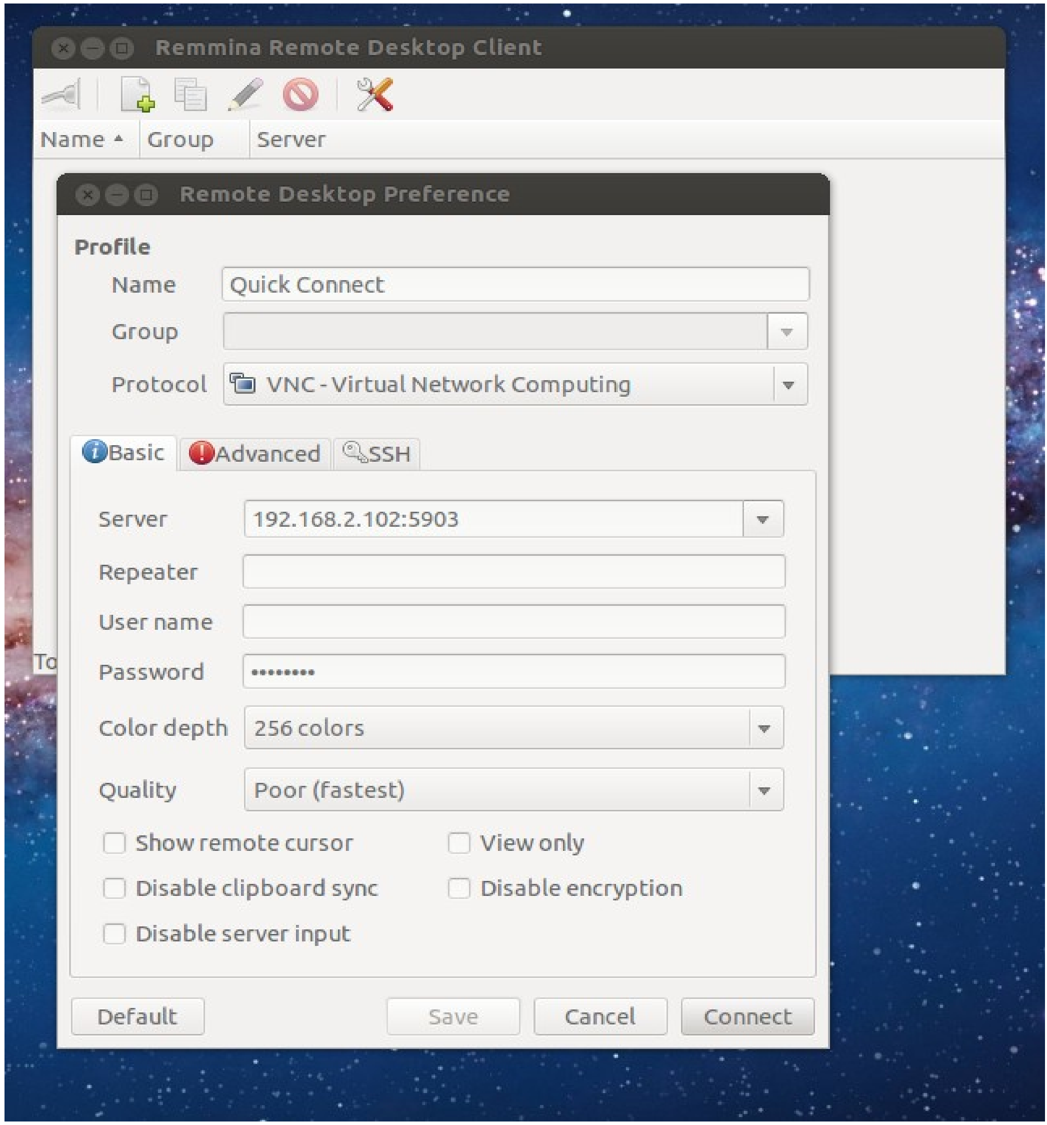
- Connect to a Raspberry Pi over VNC from Ubuntu
- On the Raspberry Pi
  - ***sudo apt-get install tightvncserver***
  - To start the server: ***vncserver :3***
    - set a password
  - To stop the server: ***vncserver -kill :3***
  - Use ***ifconfig*** command to get the IP of your RPI
- On the Desktop (running Ubuntu)
  - Open up Remmina



VNC continued

- Setup the connection in Remmina
  - A new connection
  - Protocol should be VNC
  - Enter the RPi's IP
  - The port is 5903

- Use ***Toggle to Full Screen*** option to set display size



Remmina Remote Desktop Client

Remote Desktop Preference

Profile

Name: Quick Connect

Group:

Protocol: VNC - Virtual Network Computing

Basic | Advanced | SSH

Server: 192.168.2.102:5903

Repeater:

User name:

Password: ••••••••

Color depth: 256 colors

Quality: Poor (fastest)

☐ Show remote cursor ☐ View only

☐ Disable clipboard sync ☐ Disable encryption

☐ Disable server input

History

- 1969 Dennis Ritchie & Ken Thompson developed the C language and the Unix operating system at AT&T Bell Labs
- In the 1980s Richard Stallman started the GNU project
- In the 1990s Linus Torvalds developed Linux
- Today Linux is by far the most commonly used operating system in the world.
- There are many different distros

http://en.wikipedia.org/wiki/Dennis_Ritchie
http://en.wikipedia.org/wiki/Richard_Stallman
http://en.wikipedia.org/wiki/Linus_Torvalds
http://kernel.org
http://lwn.net/Articles/472852/
http://www.linuxfoundation.org/
http://en.wikipedia.org/wiki/Linux
http://www.levenez.com/unix/ (a huge Unix history poster)

Overview

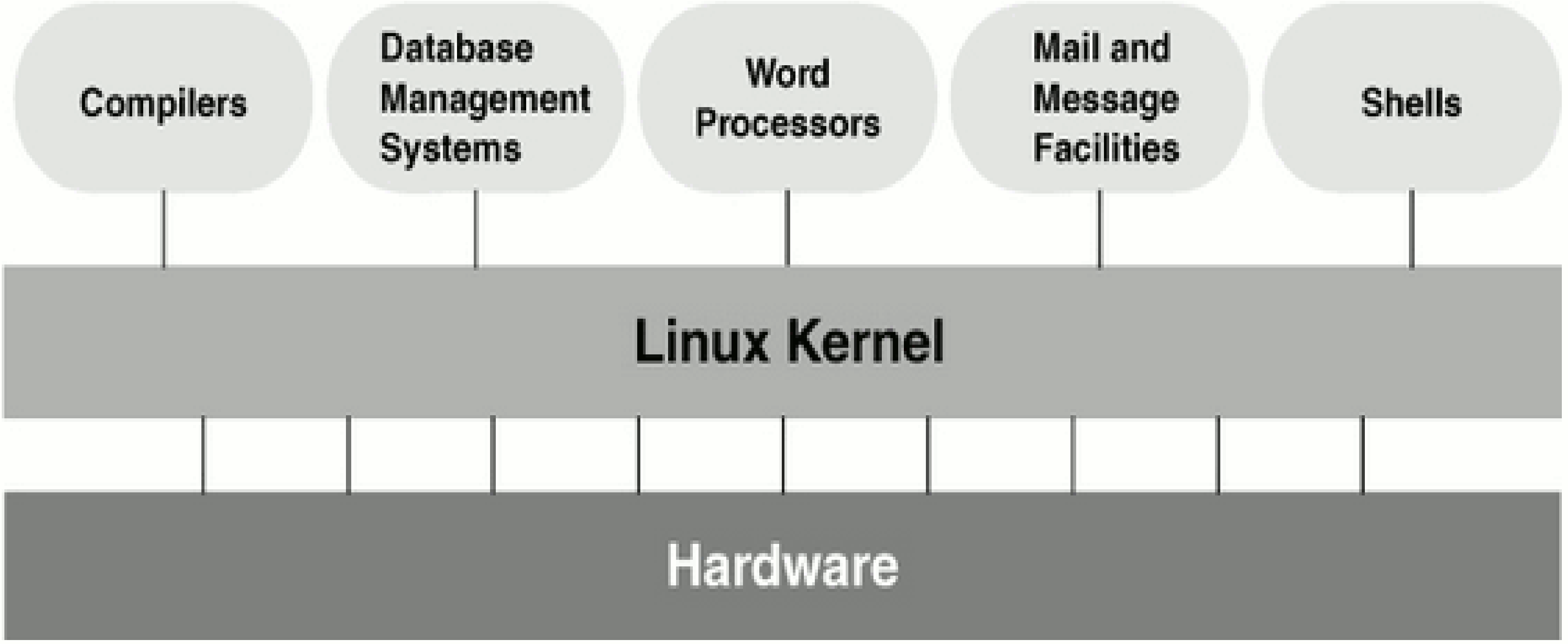


Figure 1-1. A layered view of the Linux operating system

Features

- Kernel programming interface
- Can support from 1 to more than 1,000 users
- Each user can run many processes at a time. Processes can communicate with one another but are protected from one another.
- Provides a secure hierarchical filesystem

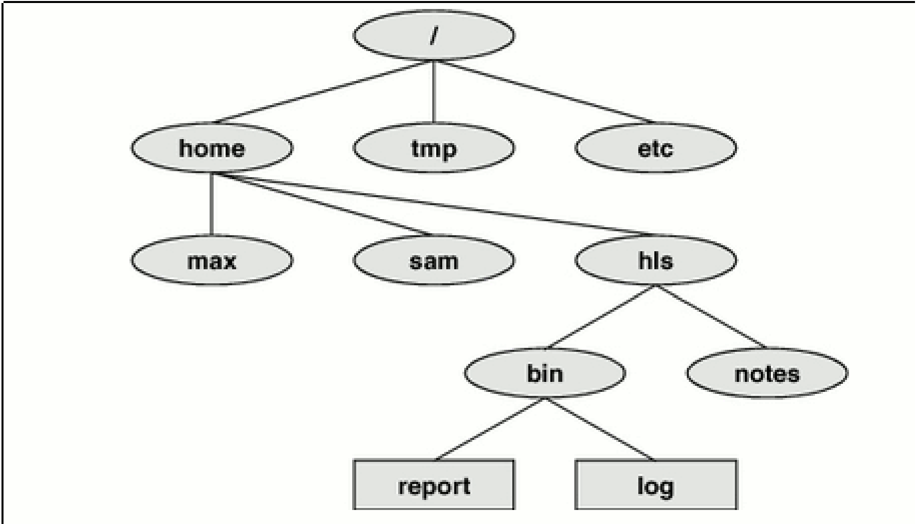


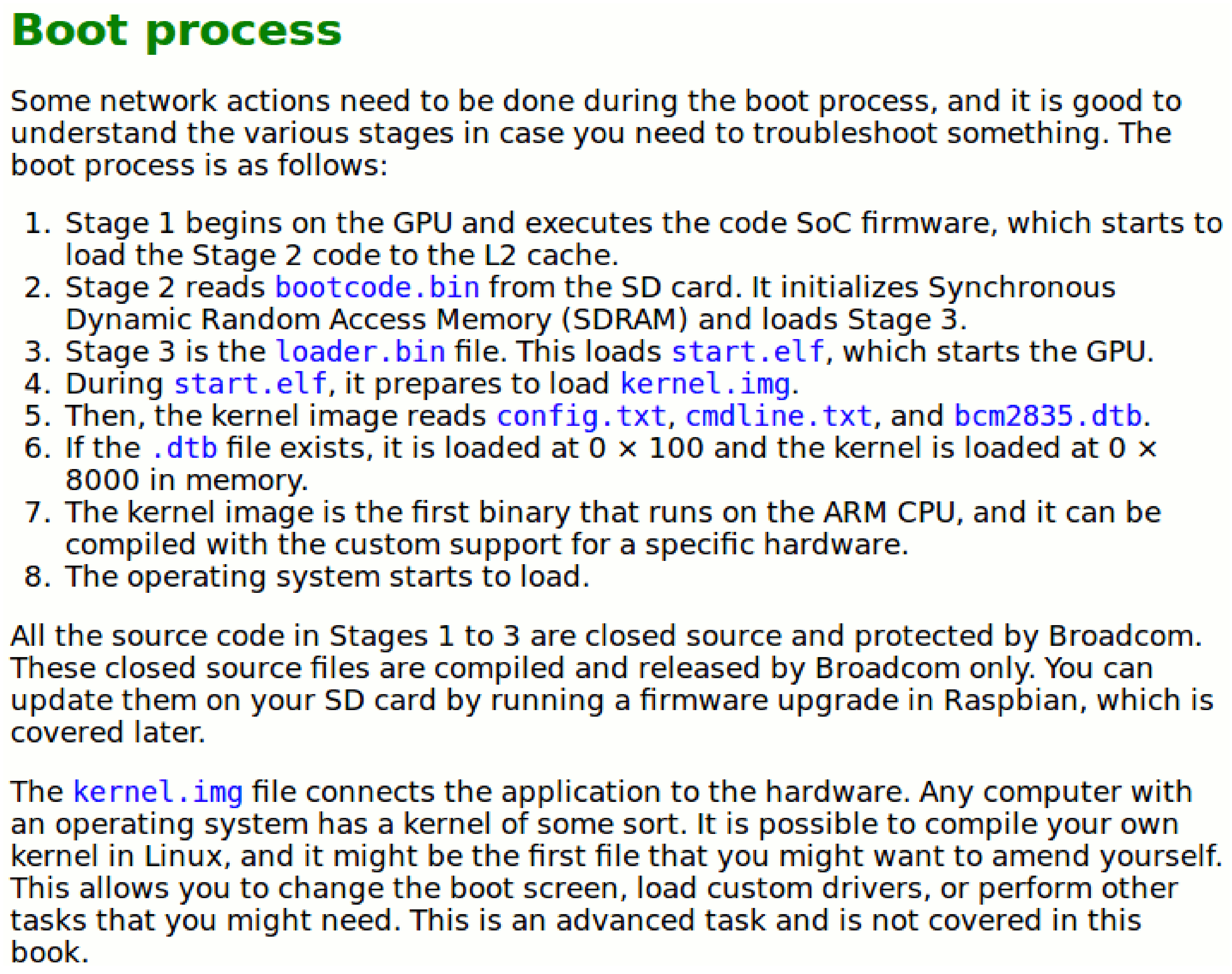
Figure 1-2. The Linux filesystem structure

Features (continued)

- The Shell: command interpreter and programming language
  - Filename generation & completion
  - Device-Independent input and output
  - Job control
  - A large collection of useful utilities (i.e., commands)
- Interprocess communication
- System administration
- GUIs
- Networking utilities
- Supports compilers and interpreters for many computer languages

Start at the Beginning

(from Raspberry Pi Server Essentials)



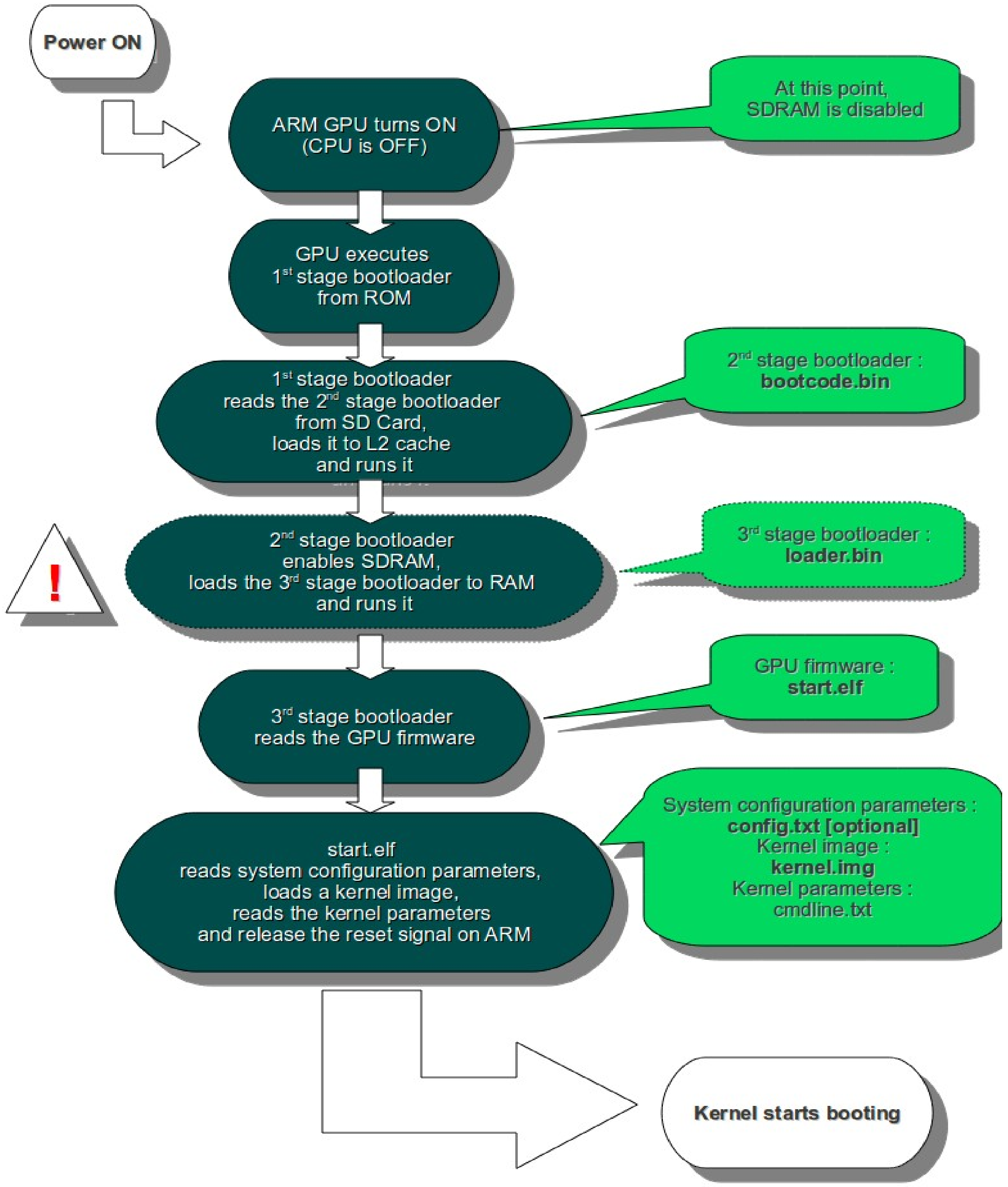
Boot process

Some network actions need to be done during the boot process, and it is good to understand the various stages in case you need to troubleshoot something. The boot process is as follows:

1. Stage 1 begins on the GPU and executes the code SoC firmware, which starts to load the Stage 2 code to the L2 cache.
2. Stage 2 reads `bootcode.bin` from the SD card. It initializes Synchronous Dynamic Random Access Memory (SDRAM) and loads Stage 3.
3. Stage 3 is the `loader.bin` file. This loads `start.elf`, which starts the GPU.
4. During `start.elf`, it prepares to load `kernel.img`.
5. Then, the kernel image reads `config.txt`, `cmdline.txt`, and `bcm2835.dtb`.
6. If the `.dtb` file exists, it is loaded at 0 × 100 and the kernel is loaded at 0 × 8000 in memory.
7. The kernel image is the first binary that runs on the ARM CPU, and it can be compiled with the custom support for a specific hardware.
8. The operating system starts to load.

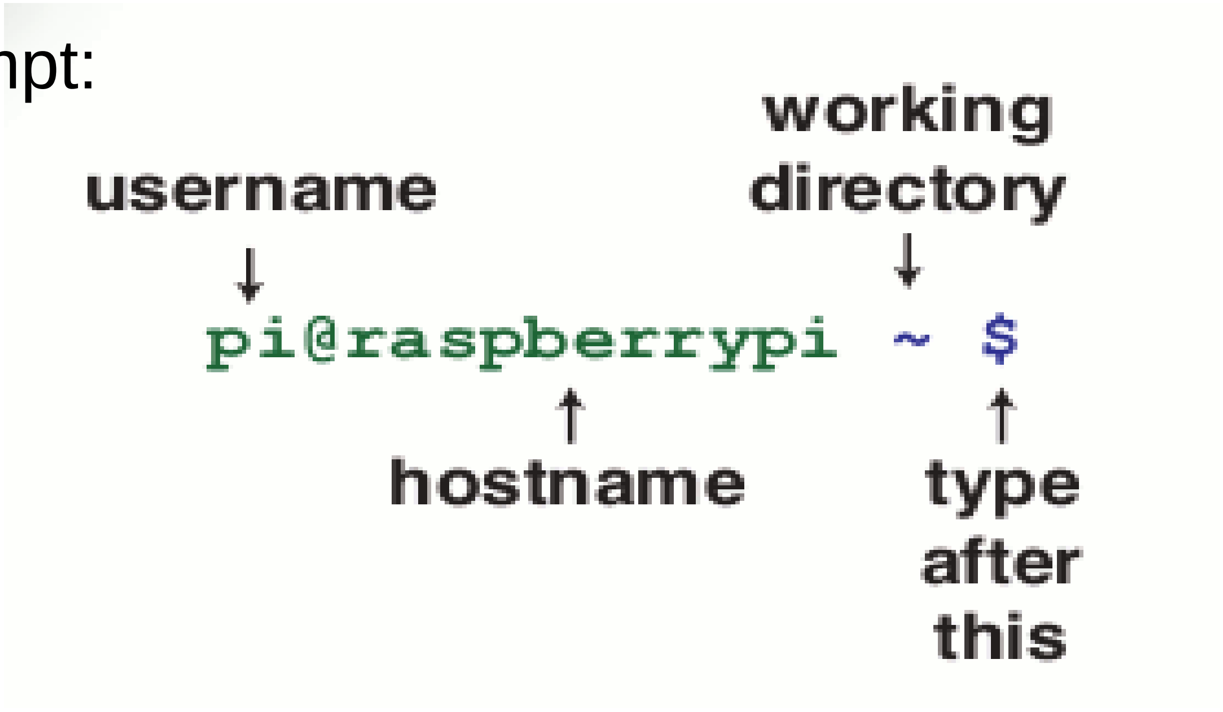
All the source code in Stages 1 to 3 are closed source and protected by Broadcom. These closed source files are compiled and released by Broadcom only. You can update them on your SD card by running a firmware upgrade in Raspbian, which is covered later.

The `kernel.img` file connects the application to the hardware. Any computer with an operating system has a kernel of some sort. It is possible to compile your own kernel in Linux, and it might be the first file that you might want to amend yourself. This allows you to change the boot screen, load custom drivers, or perform other tasks that you might need. This is an advanced task and is not covered in this book.


Power ON
ARM GPU turns ON (CPU is OFF)
At this point, SDRAM is disabled
GPU executes 1st stage bootloader from ROM
1st stage bootloader reads the 2nd stage bootloader from SD Card, loads it to L2 cache and runs it
2nd stage bootloader : **bootcode.bin**
!
2nd stage bootloader enables SDRAM, loads the 3rd stage bootloader to RAM and runs it
3rd stage bootloader : **loader.bin**
3rd stage bootloader reads the GPU firmware
GPU firmware : **start.elf**
start.elf reads system configuration parameters, loads a kernel image, reads the kernel parameters and release the reset signal on ARM
System configuration parameters : **config.txt [optional]** Kernel image : **kernel.img** Kernel parameters : cmdline.txt
Kernel starts booting

The command line

The prompt:



A command:

```
$ echo “Hello World”
```

The Shell

- **Shell** : a program that prompts the user for commands, accepts them, executes them and displays the results. A shell is just a program running on the Linux operating system like any other program
  - The shell program is a process that’s waiting for you to type something at the keyboard (standard input)
  - When you pressed enter after having typed your command, the shell interprets your input, and starts a second process to run the echo program for you.
  - It handles access to the keyboard and monitor (standard output) over to the process running echo, and then goes to sleep briefly to wait for the command to finish.
  - When the command finishes, the shell wakes up again, and takes back control of the keyboard/screen (standard input/output)
  - There are many “shells”
    - Raspbian (Debian) employs the Bourne Again Shell (Bash) by default

A few commands

- ls
- rm
- cat
- cp, mv
- pwd
- cd
- rmdir, mkdir
- grep
- head, tail
- more, less
- script
- man

Piping:

```
ls | wc -w
```

Redirection:

```
echo “hi” > filename
```

Make a new command:

```
alias l="ls -al | more"
```

Create more complex commands using a ***script***

File Permissions

- Type : ls -l
  - Linux shows detailed information, with permissions at the left.
  - Each permission is listed as rwx for user, group, and all, in that order. The first letter on the left is d if the file is a directory.
  - a - indicates a permission is not available.
- Type chmod a = rwx <*filename*> and then ls -l
- Try chmod o=r <*filename*>   followed by ls -l
- Try chmod 644 <*filename*> followed by ls -l

History and Tab completion

- Hitting the ***`tab`*** after a partially typed command will (attempt to) auto-complete the command
- The ***`up-arrow`*** recalls the previous command
- Keep pressing ***`up-arrow`*** to scroll back through your command history. The down-arrow moves forward through the history
- Press ***`right-arrow`*** and ***`left-arrow`*** to move the cursor on the command line.
  - Type to insert text at the cursor position
- Type ***`history`*** to display a list of your previous commands in order, with an index
- Type ! followed by a number with no space and press ***`enter`*** to use an index number to select a command.
  - Linux runs the command you select.

Linux File System

- boot—This contains the Linux kernel and other packages needed to start the Pi.
- bin—Operating system-related binary files, like those required to run the GUI, are stored here.
- dev—This is a virtual directory, which doesn't actually exist on the SD card. All the devices connected to the system—including storage devices, the sound card and the HDMI port—can be accessed from here.
- etc—This stores miscellaneous configuration files, including the list of users and their encrypted passwords.
- home—Each user gets a subdirectory beneath this directory to store all their personal files.
- lib—This is a storage space for libraries, which are shared bits of code required by numerous different applications.
- lost+found—This is a special directory where file fragments are stored if the system crashes.
- media—This is a special directory for removable storage devices, like USB memory sticks or external CD drives.
- mnt—This folder is used to manually mount storage devices, such as external hard drives.
- opt—This stores optional software that is not part of the operating system itself. If you install new software to your Pi, it will usually go here.
- proc—This is another virtual directory, containing information about running programs which are known in Linux as processes.
- selinux—Files related to Security Enhanced Linux, a suite of security utilities originally developed by the US National Security Agency.
- sbin—This stores special binary files, primarily used by the root (superuser) account for system maintenance.
- sys—This directory is where special operating system files are stored.
- tmp—Temporary files are stored here automatically.
- usr—This directory provides storage for user-accessible programs.
- var—This is a virtual directory that programs use to store changing values or variables.