# Summary of the Subject

Nick Urbanik

nicku@nicku.org

A computing department

## Contents

## Open Protocols and Open Standards

## Open Standards do not limit access

- Data encoded in a proprietary format may be expensive to recover far into the future
- Legal restrictions imposed by patents may require additional royalties to be paid in addition to the costs of reverse-engineering.
- See the updated notes on Free Software and Open Standards.

## Operating System Types

## Four Structures

- We covered four OS structures:
  - Monolithic
  - Layered
  - Microkernel
  - Virtual Machine

  **Monolithic** OS: examples: Linux, some Unix systems. All kernel code executes in the same address space—low communication overhead

  **Layered** Attempts to isolate parts of OS from each other to make the system more modular; has increased overhead of communication between the layers

  **Microkernel** tries to make the OS kernel as small as possible. Overhead of communication between the many simple components makes it hard for anyone to understand the system.
- Make sure you know what a *system call* and a `trap` are.

## Virtual Machine

- IBM sell many mainframes
  - very large, reliable, expensive computers with high input, output capability
  - Run many *virtual machines* on the one physical machine
  - Each virtual machine is isolated from the others, so virtual machines can be set up on the one mainframe for two companies that are competitors
    - No company can directly find out what is on the other virtual machines
  - One mainframe can replace many smaller servers in a data centre.

## Why mainframe better than servers?

- A company can choose whether to pay for a single mainframe or a number of separate server machines to provide their network services
- The mainframe may cost less than an equivalent number of individual servers because:
  - The load can be shared among all the virtual machines, and the mainframe CPU can be used effectively
  - Individual servers need to have enough CPU processing power to meet peak demand, but normal traffic will be much less than the peak.
  - Because of this, the individual servers will have a lot of unused processing power.
  - The mainframe will use much less floor space, and so save money
  - The mainframe will use much less electricity than the

## Shell Programming

## Shell Programming

- Make sure you understand what you are doing in the shell assignment.
- Understand how to use the `keychain` program with your assignment.
- **Note:** I have updated the pages about `keychain` in the notes in Module 13.

## POSIX Commands

## POSIX

- POSIX is a standard, which defines a standard set of system calls, a standard set of commands, and a standard shell programming language.
- Linux aims to be compliant with the POSIX standards. Many Unix systems are POSIX compliant.

## diff

- Often used like this:
  `$ diff -u ⟨orignal file⟩ ⟨new file⟩`
- Output of the `diff` command shows the differences between two sets of files.
- Output is per line:
  - if a line in ⟨*original file*⟩ is not in ⟨*new file*⟩, the output will have a '−' at the start of the line.
  - if a line in ⟨*original file*⟩ is in ⟨*new file*⟩, but not ⟨*original file*⟩, the output will have a '+' at the start of the line.
  - if a line has changed, even by one character, the line from ⟨*original file*⟩ will have a '−' in the output, while the line from ⟨*new file*⟩ will have a '+'.
  - Two or so lines are shown around the changes, so that it is easy to see where the change is. These *context lines* do not have any a '+' or '−' in front, but a space ' ' instead.

## find, xargs

- These two tools often are used go together.
- Make sure you understand how `xargs` works.

**find** uses logic expressions to find files that match particular requirements.

**grep** used to search for strings in *files* . . .

and also in standard output.

## Files and File Permissions

## File Permissions and Symbolic Links

- Make sure that you have worked though and *understood* all the problems in the Permissions Tutorial `http://nicku.org/ossi/lab/permissions/permissions.pdf`
- We have covered permissions in more detail than in previous years, and permissions are a vital topic in managing POSIX systems.
- We also spent some time studying *symbolic links*
  - Make sure you understand clearly the difference between a *relative* symbolic link and an *absolute* symbolic link
  - Make sure you understand how to create them from any directory.
  - Please study the handout about symbolic links `http://nicku.org/ossi/lab/sym-link/sym-link.pdf`

## Processes

## Processes and Threads

- Processes have a *Process Control Block* (PCB)
- A PCB is one entry in the process table
  - In Linux, it is called `task_struct`. Some people call it a *task descriptor*
- A PCB holds a lot of information, including:
  - The Process ID, (PID), PID of parent (PPID)
  - various User IDs, (UIDs), group IDs (GIDs)
  - An environment (containing environment variables such as `PATH`
  - A copy of the CPU registers the last time the process was suspended, including a copy of the program counter.
  - The process state (see the two diagrams of process state)
  - Address mapping details
  - Resources held by the process, such as a list of files

## Signals and IPC

## How Processes can Talk to Each Other

## Signals and IPC

- Processes cannot easily share information
- Need to use Inter Process Communication (IPC) for two processes to share data.
- Examples:
  - Pipes — you used in shell programming
  - Sockets — over a network (e.g., for the Internet), and through a socket file — the `ssh-agent` talks to `ssh`, `scp` and other SSH clients through a *socket*
  - Signals — See the assignment and the `trapall` shell script
- Signal is sent by the `kill()` system call
  - The `kill` shell command also makes the `kill()` system call
- A process often terminates when it recieves a signal
- A process can *trap* a signal by executing some code

## Job Control

## Job Control

- We *stop* a process with $\boxed{\text{Control-Z}}$
- This sends a `STOP` signal to the process.
- A stopped process is forced to stop executing, but is still using memory and holding resources and file locks, that it was holding when you sent it the `STOP` signal.
- Understand what `fg`, `bg`, `jobs` do.
- Read about this again in module 2.