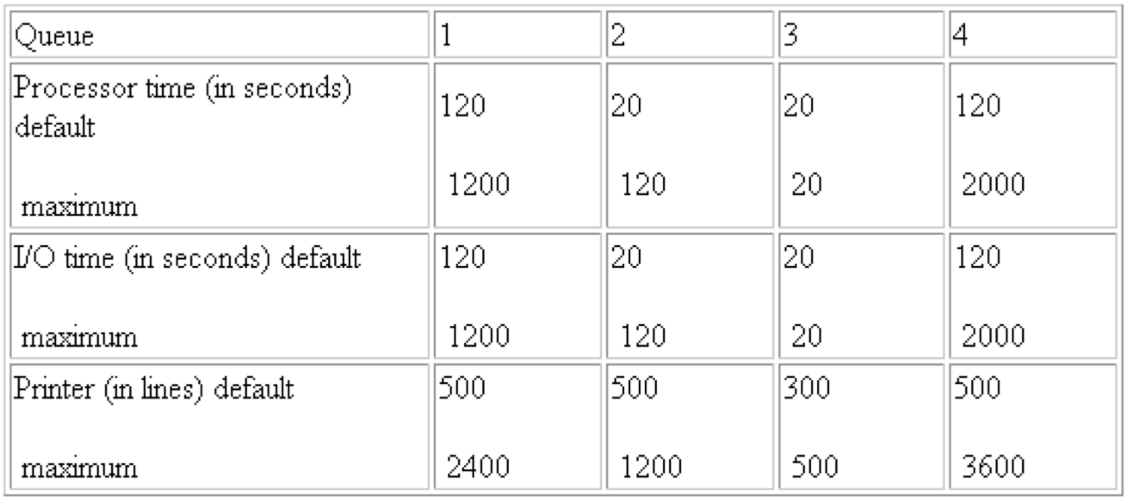
Scheduling

To maximize the use of the computing machinery the OS needed to know:

- what and how many devices a process would use
- what and how many files a process would use
- how much output it was likely to produce
- how long it was expected to take

Some of this information was submitted on the control cards, e.g. the files to be used. Some of this information was inferred by the job queue the job was submitted on. Different queues meant different expected resource requirements. Jobs on different queues were scheduled differently.

Scheduling (cont.)



Queue	1	2	3	4
Processor time (in seconds) default	120	20	20	120
maximum	1200	120	20	2000
I/O time (in seconds) default	120	20	20	120
maximum	1200	120	20	2000
Printer (in lines) default	500	500	300	500
maximum	2400	1200	500	3600

This system was used at the University of Auckland.

The user could specify limits up to the maximum allowed for a queue. If no limit was specified the default for that queue was applied.

- Queue 3 was the default. It was designed for jobs which were quite small and which would move through the system smoothly. They were not allowed to use magnetic or paper tape (hence they did not require operator intervention).
- Queue 2 jobs were allowed to use two tapes (input or output).
- Most large jobs went through Queue 1.
- Queue 4 was for very large jobs and these jobs were scheduled manually.

To stop users abusing the system, they were limited to two jobs in any one queue at a time.

Power to the people

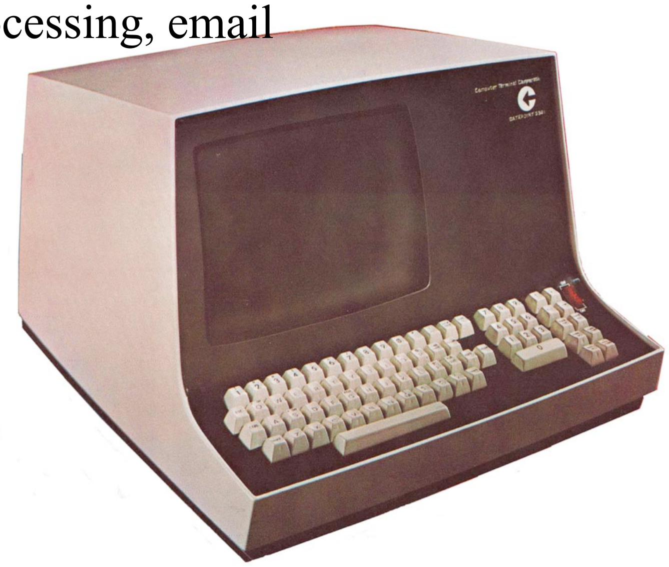
Circa 1970s.

Hardware was cheaper. This meant two things:

- Computer systems could afford to be nicer to users.
- People were now the expensive bit, we wanted to make them more productive.

Give them all a console

- CRT VDUs, early versions used teletypewriters.
- The early ones still looked like printers.
- Text was displayed on the screen top to bottom / left to right.
- You couldn't easily edit in place.
- Flushing the whole screen at 2400 baud was a bit obvious.
- Side effect - people do different things with computers
- Shock, horror! Word processing, email



Time-sharing systems

People want good response

- waiting over 1/5 of a second is noticeable
- waiting over 5 seconds is unacceptable

People working at the machine are unpredictable

- they want to use files and devices at any time
- it is very hard for the OS to work out what processes will be compute intensive or IO intensive
- more processes are active in the system

there is usually at least one process to deal with commands for each user, plus the normal work they requested

- the terminal is used for the entering of data and programs as well as JCL type commands
- because people are slow peripherals many more had to be on-line than jobs needed to be running on a batch system to get through the same amount of work
- users would also communicate with each other - early email systems and talk programs

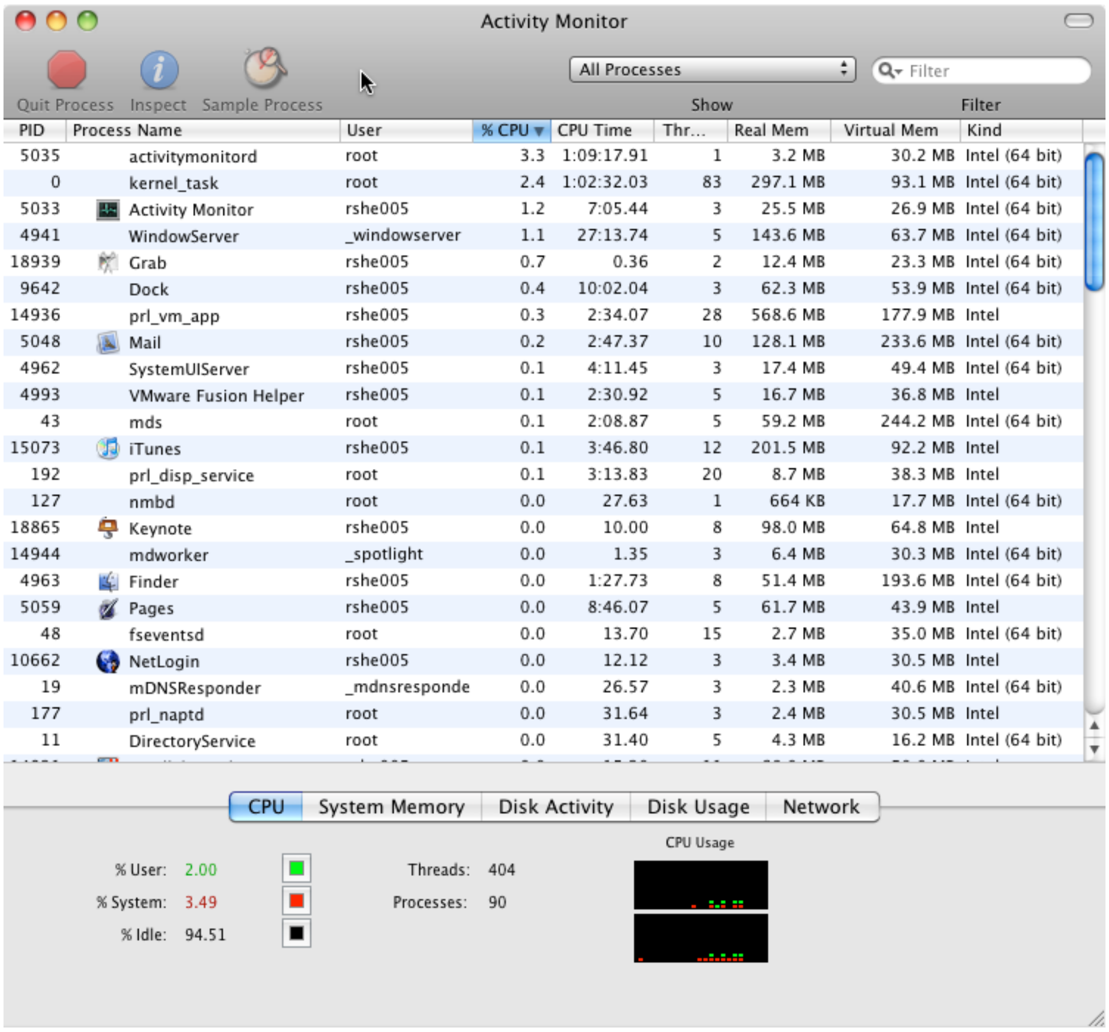
Time-sharing systems (cont.)

Devices and the CPU can’t be used to capacity

otherwise the response time is too long.

So there has to be slack in the system.

- •look at the processor usage on your Windows PC using `System monitor` or `Task Manager`
- •or run `top` on UNIX



PID	Process Name	User	% CPU	CPU Time	Thr...	Real Mem	Virtual Mem	Kind
5035	activitymonitord	root	3.3	1:09:17.91	1	3.2 MB	30.2 MB	Intel (64 bit)
0	kernel_task	root	2.4	1:02:32.03	83	297.1 MB	93.1 MB	Intel (64 bit)
5033	Activity Monitor	rshe005	1.2	7:05.44	3	25.5 MB	26.9 MB	Intel (64 bit)
4941	WindowServer	_windowserver	1.1	27:13.74	5	143.6 MB	63.7 MB	Intel (64 bit)
18939	Grab	rshe005	0.7	0.36	2	12.4 MB	23.3 MB	Intel (64 bit)
9642	Dock	rshe005	0.4	10:02.04	3	62.3 MB	53.9 MB	Intel (64 bit)
14936	prl_vm_app	rshe005	0.3	2:34.07	28	568.6 MB	177.9 MB	Intel
5048	Mail	rshe005	0.2	2:47.37	10	128.1 MB	233.6 MB	Intel (64 bit)
4962	SystemUIServer	rshe005	0.1	4:11.45	3	17.4 MB	49.4 MB	Intel (64 bit)
4993	VMware Fusion Helper	rshe005	0.1	2:30.92	5	16.7 MB	36.8 MB	Intel
43	mds	root	0.1	2:08.87	5	59.2 MB	244.2 MB	Intel (64 bit)
15073	iTunes	rshe005	0.1	3:46.80	12	201.5 MB	92.2 MB	Intel
192	prl_disp_service	root	0.1	3:13.83	20	8.7 MB	38.3 MB	Intel
127	nmbd	root	0.0	27.63	1	664 KB	17.7 MB	Intel (64 bit)
18865	Keynote	rshe005	0.0	10.00	8	98.0 MB	64.8 MB	Intel
14944	mdworker	_spotlight	0.0	1.35	3	6.4 MB	30.3 MB	Intel (64 bit)
4963	Finder	rshe005	0.0	1:27.73	8	51.4 MB	193.6 MB	Intel (64 bit)
5059	Pages	rshe005	0.0	8:46.07	5	61.7 MB	43.9 MB	Intel
48	fseventsd	root	0.0	13.70	15	2.7 MB	35.0 MB	Intel (64 bit)
10662	NetLogin	rshe005	0.0	12.12	3	3.4 MB	30.5 MB	Intel
19	mDNSResponder	_mdnsresponde	0.0	26.57	3	2.3 MB	40.6 MB	Intel (64 bit)
177	prl_naptd	root	0.0	31.64	3	2.4 MB	30.5 MB	Intel
11	DirectoryService	root	0.0	31.40	5	4.3 MB	16.2 MB	Intel (64 bit)

% User:	2.00	🟩	Threads:	404
% System:	3.49	🟥	Processes:	90
% Idle:	94.51	⬛		

Time-sharing changes

Over-demand for resources can happen easily (but we hope occasionally).

High-level scheduling decisions are made by the users.

Security is more of a problem

Hacking on a batch system was difficult.

The system must have some authentication process.

More user administration.

Profiles, defaults, resource limits

Different types of users - System administrators with more rights than “ordinary” users.

Batch system remnants

Ways of running processes at specific times without user intervention e.g. `crontab`, `at` and `Task Scheduler`.

Until the arrival of cheap graphics terminals - the terminal still looked very like a card reader. e.g the `vt52`.

Command files or shell scripts - very like the control cards of a job control language.

Desktop computers

Circa 1980s

Started with resident monitor systems.

TRS-DOS

Apple DOS

MS-DOS

MacOS

File systems

Simple single-level

There was no need for security.

Only one user.

No hard disks.

Only one program ran at a time.

People usually work with one program at a time.

Gradual need for multiprogramming

Print spooling

TSRs

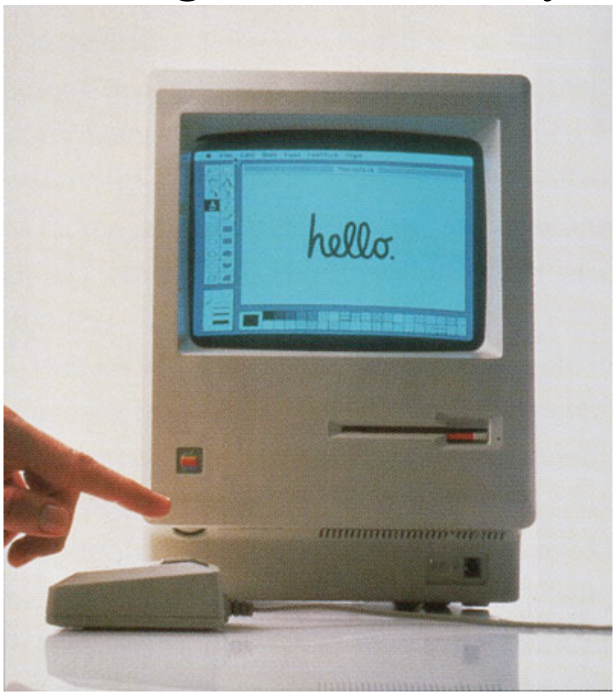
Mac Switcher

Personal computers (cont.)

Xerox (late 70s early 80s) had done some pioneering work on making computers easier to use.
This was originally on time-sharing and networked workstations.

- needed high definition graphics screens
- desktop metaphor - make the computer look like something else (even though *files* were called *files* well before this)

Macintosh (now officially called "Mac") took this and popularized a cut-down version.
MS-DOS users laughed scornfully until 1995.



Personal computers (cont.)

Eventually personal computers became as powerful as “real computers”.

Time-sharing OSs could fit on the desktop.

UNIX was the main example.

Fully featured PC operating systems were developed with

virtual memory

multiprogramming

sophisticated file systems

networking

multi-user support

e.g. MacOS X, Windows 8, Linux, FreeBSD

Networks

1980s on

New problems

security

providing transparent access to resources

developing protocols

Network Operating System

provides file sharing

provides communication scheme

runs independently from other computers on the network

Distributed Operating System

less autonomy between computers

gives the impression there is a single operating system
controlling the network.

Both are known as

loosely coupled as opposed to *tightly coupled systems*

Multiprocessor systems

Tightly coupled system – processors share memory and a clock; communication usually takes place through the shared memory.

Almost all computers now count as tightly coupled systems.

Advantages of parallel systems

Increased *throughput*

Economical for the increase in performance

Increased reliability

graceful degradation

fail-soft systems (shut down non-essentials)

Symmetric multiprocessing (SMP)

Each processor runs an identical copy of the operating system.

Many processes can run at once without performance deterioration.

Most modern operating systems support SMP

- essential now that we have multiple core CPUs

Asymmetric multiprocessing

Each processor is assigned a specific task; master processor schedules and allocates work to slave processors.

More common in extremely large systems

Real-time systems

A completely different thread of OS development.

Real-time systems have varying levels of timing constraints.

Hard real-time – must satisfy requests within definite time periods or the system fails.

e.g. robotics, air traffic control, nuclear power plants

Soft real-time – it doesn’t matter too much if a time constraint is not met exactly.

e.g. Multi-media software, telephone system

Most modern OSs can handle some sort of soft real-time control.

Hard real-time control systems have to be specially designed.

Pocket computers and smartphones

PalmOS (before WebOS) - small memory and slow processors. The OS had to be very efficient in order to get reasonable performance. No memory protection, no virtual memory.

Android – Based on Linux, programming applications is done in Java to Android specific APIs. A cut down version of Linux for embedded devices.

https://www.youtube.com/watch?v=Y6RcQ4H0lKg

iOS - Based on MacOS X. Virtual memory, includes paging for code but not for data (why?)

SymbianOS – also true memory protection.

Other problems of size and battery consumption.

GUI, amount of RAM that can be kept valid.

Before the next lecture

Read textbook sections

Chapter 16 Virtual Machines