

Operating Systems Quick Revision Notes

Why OS

- manage interaction
 - users
 - programs
 - networks
 - different hardware
- manage resources
 - Processes
 - tasks
 - jobs
 - threads
 - Memory
 - Files
 - Networks
 - Input
 - Output
- Computer has
 - Hardware
 - Software
 - System
 - Applications

What is an os

- Extended machine
- Resource Manager

Types of OS

- Mainframe OS's
- Server OS's
- Multiprocessor OS's
- Personal computer OS's
- Real Time OS's
- Embedded OS's
- Smart Card OS's

OS Structure

- Bootstrap
 - Modern PC has several
 - BIOS
 - Disk/Network
 - kernel
 - Contains
 - Device Drivers
 - File system code
 - Process Management
 - Scheduler
 - Monolithic
 - Everything in main kernel
 - Microkernel
 - Everything in separate processes
 - Layered
 - Virtual Machines
- Implementing
 - Simple Library called by apps (dos)
 - protected mode
 - Programs use traps (software interrupts to access kernel)
 - Hardware memory Management

----Could support Full Pre emptive multitasking

-Utilities

- Disk checking utilities
- Backup programs
- Printng programs
- Command interpreter
- GUI
- File System Tuning apps
- User Admin
- Queue Management
- Compilers/Linkers

Unix Onion

-Works out from kernel all the way to user apps

Interrupts

- Run in a higher mode Supervisor/Privaledged
- Can access all the memory due to above
- Registers have to be saved on entering, possibly alternative stack

Scheduling

- Dos only one program at a time (single thread with interrupts)
- Multiple co-operative threads (work together no fancy code needed)
- Full Preemptive schedular

Memory Systems

- Registers
- Cache
 - one line of data which just corresponds to the tag
 - fast and simple
 - memory competes for same space (differing index same tag) aliased (causes thrashing when moving lots of memory/vector arithmetic)
- 2nd Level cache
 - Associative cache
 - Multiple lines for each tag bit
 - Slower more complex
 - Replacement alg needed
- Ram
- Disk

Memory Management

- Fragmentation
- Paged virtual memory
- Hardware Support
- Translation
- TLB

- Basic Memory management
- Just have os and program

- Multiprogramming fixed partitions
 - Program has to fit in a partition
 - Causes Fragmentation
 - Wasted space when program leaves a bit

-Segmentation

- Dynamically create partitions of the right size
- No internal fragmentation
- External fragmentation when segments dont line up

```

-Swapping
--processing are moved in and out of memory
--?

-Virtual Memory
--MMU sits between cpu and memory and translates address's
--Paging
---All memory is split into 4k segs and each can be swapped in and out as needed
--PAGE TABLES ARGH LOADS OF RANDOM CRAP HERE!!!!1
--TLB ETC !!!!!!!!!11

--page replacement algorithms
---which must be removed
---best to choose least used/unmodified
--OPTIMAL PAGE REPLACEMENT
--NOT RECENTLY USED
--LEAST RECENTLY USED
--Page size
---Small
----less fragmentation
----better for most programs
----less wasted memory
----programs need more pages
----bad for data manipulation apps
--Shared pages
---Programs share data
--Implementing paging
---Process Creation
---Process Execution
---Page Fault Time
---Process Termination Time

--Handling page faults
---hardware traps to kernel
---General Registers saved
---Os determines required page
---Checks address
---Loads if required
---page tables updated
---Faulting instruction backed up
---Faulting instruction restored
---Registers restored
---Program runs hapily ever after

--Locking pages in memory
---Virtual Memory and IO need to interact on reall memory DMA etc

```

Advantages / Disadvantages

```

-Paging
--ADD: prgrammer doesnt need to know
--DIS: one linera address space
--ADD: can have more memory than the machine has
--DIS: data and code can't be kept seprate
-Segmentation
--DIS: Programmer needs to be aware of system
--ADD: Many sets of linear address space
--ADD: can have more memory than u have
--ADD: Data and code can be kept seprate
--ADD: fluctuating table sizes work hapily
--ADD: data can be shared between users

```

File Systems

- Understand FAT12

Processes

- Processes
- Start
- System Init
- Process create system starts
- User loads new process
- Batch job
- Stop
- Normal Exit
- Error Exit
- Fatal Exit
- Killed
- Threads

Scheduler

- Mimic kernel thread functions
- Avoids unnecessary user/kernel interactions
- Virtual processor for each process
- Problem
- Relies on kernel
- Calling procedures in user space

Synchronisation

- Hardware techniques
- Critical regions
- Semaphores
- Monitors
- Dining philosophers, readers/writers
- Avoiding Deadlock

Message Parsing

- Mutual Exclusion
- Ports mail boxes

Scheduling

- LOADS

Realtime systems

- Different priorities
- Reliability
- Fail-Safe
- Scheduling
- Rate Monotonic
- Deadline

Networks

- Network Protocols
- Sockets

Security

Virus'h hacking etc

Bla Bla had enuff