Contents

Course description

To describe the services an operating system provides to users, processes, and other systems
To discuss the various ways of structuring an operating system
To explain how operating systems are installed and customized and how they boot

Question 1 - Operating System Structures - ch 2

User interfaces = Command-Line (CLI), Graphics User Interface (GUI), Batch

Keywords

1.) Short introduction
explain the 3 types of interfaces. GUI, CLI, and batch
Explain pros and cons
CLI - Requires more knowlage about how it works.
GUI - More intuative and easier to use

2.)
How you communicate between OS and programs,
System calls - How do you call them, how to use them. parameters etc.
Check file with the list of system calls
Find the file corresponding to the system call.
puts the parameters
using registers
stores in block and address of block is passed to the function as a parameters
places or pushed on the stack and POPed by the operating system.

3*) Process communications
Message parsing and shared memory
pro/cons
shared memory, uses less space as memory is only copied if it gets changed.
Message parsing
Either directly between two processes via mailbox handled by the os

4*) OS structure
Simple structure
There is no control, applications can talk directly to hardware and the kernel.
Layered
There are multiple layers,the user can only talk to the the applications,the applications can only talk to the operating system API, and only the operating system can call to the kernel
User->Application->OS->Hardware
Micro-kernel
Move as much as the kernel as you can into user-space. User mode applications can communicate via the kernel.
Modules
Indivisdual modules that can be loaded and unloaded into the kernel, so provide some kind of use.
Hybrid-systems
Most modern operating systems are not one pure model,

5) Debugging
What can we do when something goes wrong?
Log files, data dumps, tracing tools. Windows task-manager essential, Linux has the top/h-top command

6) Boot-loaders, How does the os know where to find the boot-loader, kernel, etc.
Multiple stages, Bios loads a 512(less due to constraints)boot loader into memory from the master boot record on a disk and jumps to it. this then either loads a secondary bootloader, or the kernel.

Questions from lecture

What is a system call.
Programming interface to the services provided by the operating system

What groups of system calls are there.
File manipulation, process control, security and protection, communication, I/O device manipulation & information management

why would you use a system call API.
Ease of use, Reduce failure, reuseability/portability

What ways are there to pass parameters to the OS.
Stack + reference, register, memory -> send reference

How can an operating system be structured(architecture wise)
Layered structure, module based, micro-kernel & hybrid systems

what code is first read when you turn on your computer
Bootstrap code located in EPROM

Question 2 - Process Concept and Multi threaded Programming - ch 3 & 4

Keywords

1) Introduction - Process control block.
Diagram on slide 3.7
- What is a process.
- Process control block.
  - State
  - Process number
  - Program counter
  - Registers
  - Memory limits
  - Files descriptors (active files)
  - ....

2) Process scheduling -
- CPU & I/O burst
- Short, medium & long schedulers.
  - Short - ready, running and waiting.
  - Medium - Swaps Out to disk.
  - Long - Loads jobs into the new queue from a work queue.

3) Context switch.
- Saves current state of the process and stores it, when the process waits for I/O.

4) Interprocess communication -
- Message parsing
  - The processes can communicate with each other without resorting to shared variables
  - This is done using by using a IPC facility that has two operations. a send and a receive operation.
  - If eg. P and Q wish to communicate they'll need to first establish a link between them, and then exchange the message via send/receive.
  - Direct and indirect(mailbox with id)
- Shared memory
  - Fixed shared memory segment
  - The processes have to check if the memory changed.

5) Communication-Client/server
- RPC (Remote Procedure Call)
  - Abstract procedure calls on networked systems
- Socket
  - pair of sockets using TCP/UDP - Ip:port
- Pipes
  - Unamed pipes - Unidirectional, and require parent child relation
  - Named pipes - bidirectional, and can be used by multiple processes.

6) Multitreading models
- Many to one
- One to One
- Many to Many
- Hybrid - 2 level model

7) Multicore programming

Amdahl Law - S is serial portion and n is processing cores.

$$speedup = \frac{1}{S + \frac{1-S}{N}} \tag{1}$$

8) Concurrency /parallel -
Whats the difference
Concurrency
One program using multiple threads doing different things. eg, consumer/producer
parallel
A program using multiple threads to do the same work.

9) Thread libraries
POSIX
Pthread
javathread
Winthread(win32)

10) Implicit Threading
Thread pools
OpenMP - Compiler finds code that can be run in parallel and compiles it so it is.
Grand Central dispatch - Allows identification of parallel sections, manages most of the details.
two types of queues, serial where its FIFO and concurrent. removed in FIFO but multiple items can be removed at the same time.

11) Threading Issues
Signal handling
Thread Cancellation - When should a thread be terminated
Thread-Local Storage - Each thread had its own copy of the data(waste of space)
Scheduler activations

Questions from lecture

What state can a process by in?
- New.
- Ready.
- running.
- waiting.
- terminated.

What is a process control block?
- A block where the OS store information about a program.
- Used for content switching and stores the following information:
  - Process state
  - Process number
  - Process counter
  - Registers
  - Memory limits
  - List of open files

Describe ways to do IPC
- Message parsing and memory shearing.

What is the difference between a process and a thread
- A process is a thing we need to execute, and thread is a subtask of a process.

What advantages is there when using threads?
- Threading lets you work on multiple cores at the same time, therefor letting you run a job faster.

Difference between parallelism and concurrency?
- Parallelism - same task spread on multiple threads
- Concurrency - program running with multiple threads
- We can have Concurrency without Parallelism but not the other way around

What are the most common API's for user level threads
- Boost (maybe builds on pthread)
- Pthread
- Java thread
- Winthread

What are the implicit threading methods.
- Thead pools
- Openmp
- Grand Central Dispatch

Question 3 - Process Scheduling - ch 5

Scheduling types
FCFS - unstable wait time.
Shortest job first. optimal. gives minium wait time, but only usable for long time sheduling.

Keywords

1. Introduktion
   what is CPU bound
   what is I/O bound
   explain as shortterm, medium term, and long term
2. Criterias
   What do we want to take a look at as I/O bound, like what can we do to get a better CPU Util
3. Algorithmer
   Explain FCFS, SJF, etc.
   Schedulers.
   First come first service
   Shortest job first
   Shortest time remaining first
   Round robin
   give each process the same amount of time and loop over them until they're done. circular queue
   starvation etc.
4. Multi processer scheduling.
   Single queue, queue per core, shared queues
5. Realtime schduling
6. Examples

Questions from lecture

1. When is it Relevent for the scheduler to take decision.
    When a new process queued, and a new processes is started.
    4 cases
        when a process running state to waiting.
        when a process terminates.
        When a process is queued
        When a process goes from waiting to ready

What is dispatch latency.
    time it takes for a interupting process to become active
    move memory and registers from the running task, move new tasks there and start it

What scheduling ctiteria can we use.
    maximize CPU util, low latency for take critical jobs jobs.
    runtime, priority, CPU bound, I/p bound

Describe the scheduling algortirhms: FCFS, SJF, Shortest remaining time fist, RR.
    First come first servce
    Shortest job first
    Shortest time remaining first
    Round robin
    give each process the same amount of time and loop over them until they're done. circular queue

Describe priority schduling and aging
    The longer it has been in the queue the higher the priority the job has

What is the difference betweeen asymmetric and symmetric multiprocessing
    Asymmmetric means tasks wont wait for other things
    Not all processes has the some capapicities
    All the kernals can do the same.

What is a memory stall
    Waiting for memory ?

What is the difference between soft and hard real-time systems.
    Soft - Don't time life crititial systems while hard real time systems have.

Describe rate montonic scheduling and earlist deadline scheduling
    rate montonic scheduling - Schedule jobs in intervals, like a, b, c.
    earliest deadline scheduling - Schedule the process with the first deadline first.

Question 4 - Synchronization - ch 6

Keywords

1. Introduction
2. Critital section
   What is it?
   - The 3 requirements: mutual exclusion, progress, bounded waiting (Software Solution)
   - Peterson's solution as example in book, but not too useful to cover

3. software solution to critical section

4. hardware solutions to critical section
   Why is hardware a better solution than software?
   'test_and_set()' and 'compare_and_swap()

5. mutex lock
   Simplest
   Check a variable if we can enter section
   Setting/checking variables are NOT atomic actions so we need to use hardware help

6. semaphore lock
   More complex solution
   Allows for more customization
   Such as a MAX of processes at once

7. monitors
   A lock that can release its locks again if all locks needed can't be acquired, this can be used to solve the dining philosophers problem.

8. spinlock
   When do we want to use spin lock, and when do we use Mutex locks instead

9. examples

10. alternative solution
    Usage of library such as OpenMP, which us define pragmas instead and functional programming languages

Questions from lecture

Describe the terms "race condition" and "Critical section"

When two proceses want to access a shared resource. that may be unstable if its done in a non-locked way
Race condition is when multiple processes/threads are competing about the same resources and a undesired output may happen.
Critical section is a section of code that sensitive to race conditions.

What should a solution to the critical section satisfy

Mutual exclution, progress, and
that only one thread can be active in the critical section at the time.

What is preemptive vs. non preemptive

preemptive is the act is temporarily interrupting a process
non-preemptive - When a process enters the state of running, the state of that process is not deleted from the scheduler until it finishes its service time.

Describe "test and set" and "compare and swap".

Two different ways of implementing a mutex lock, that are garenttted to be excuted in an atomic way.
Test and set -
Compare and swap can only be used by one function, its checks for a codition and sets swaps two value if the condition is met.

What is a mutex and a semaphore

a Mutex lock is a lock where only the process holding the lock is allowed to perform actions in the locked section.
A mutex contains a boolean and is acquired and released.
A binary semaphore is the same as a mutex.
a counting semaphore is when the lock uses a counter if multiple processes can be active within the lock.

Describe some classic problems of synchronization.

Readers writers problems.
dining philosopher problem.

What is a monitor

We can put functions into the monitor and only one process can be active within the monitor at the time, else the process will have to wait in the queue.

Question 5 - Deadlocks - ch 7

Keywords

1. Introduction
    - What is a deadlock
    - 4 conditions
        - mutual exclusion, hold and wait, circular wait, and no pre-emption
        - Show a small example, resource graph

2. How to handle a deadlock
    - Prevent and avoid
    - recover
        - Start killing tasks that are in the deadlock until its resolved
    - Ignore
        - Let the user handle it

3. How to avoid a deadlock
    - How to avoid the 4 conditions
        - If we can change one we can avoid

4. avoiding a deadlock
    - bankers algorithm
    - resources graphs
    - what is the criteria for that they are in safe and unsafe states

5. Deadlock detection
    - resources graph detection

6. why do we want to avoid deadlocks
    - give examples on when to avoid deadlocks.

7. Recover
    - Show some algorithms on how to avoid a deadlock

8. Examples

Notes

Objectives

To develop a description of deadlocks, which prevent sets of concurrent processes from completing their tasks
To present a number of different methods for preventing or avoiding deadlocks in a computer system

What is a deadlock

4 things have to hold for a deadlock to happen, these are Mutual exclusion Hold and wait No preemption Circular wait
Mutual exclusion
: Only one process at a time can use a resource

Hold and wait
: A process holding at least one resource is waiting to acquire additional resources held by other processes

No preemption
: A resource can be released only voluntarily by the process holding it, after that process has completed its task

Circular wait
: There exists a set P0 , P1 , … , Pn of waiting processes such that P0 is waiting for a resource that is held by P1 , P1 is waiting for a resource that is held by P2 , … , Pn–1 is waiting for a resource that is held by Pn, and Pn is waiting for a resource that is held by P0.

  We have a circle of tasks waiting for each other.

How can a deadlock be detected

Graph detection

Deadlocks can be detected by looking for cycles in a graph, this doesn't mean that we have a deadlock, but it means we have a risk of there being one.
no cycles → no deadlock If graph contains a cycle →
: if only one instance per resource type, then deadlock
  if several instances per resource type, possibility of deadlock

Methods for handling deadlocks

1. Ensure that the system will never enter a deadlock state
2. Allow the system to enter a deadlock state and then recover
3. Ignore the problem and pretend that deadlocks never occur in the system;
   used by most operating systems

Deadlock prevention

Mutual exclusion -
: not required for sharable resources; must hold for nonsharable resources

Hold and wait -
: must guarantee that whenever a process requests a resource, it does not hold any other resources.
  Request all resources up front before execution. (all or none)

No Preemption -
: If a process that is holding some resources requests another resource that cannot be immediately allocated to it, then all resources currently being held are released
  Preempted resources are added to the list of resources for which the process is waiting
  Process will be restarted only when it can regain its old resources, as well as the new ones that it is requesting

Circular Wait
: Impose a total ordering of all resource types, and require that each process requests resources in an increasing order of enumeration.

Deadlock avoidance

Requires that the system has some additional a priori information available
Simplest and most useful model requires that each process declare the maximum number of resources of each type that it may need

The deadlock-avoidance algorithm dynamically examines the resourceallocation state to ensure that there can never be a circular-wait condition
Resource-allocation state is defined by the number of available and allocated resources, and the maximum demands of the processes.

Avoidance algorithms

Single instance of a resource type $\rightarrow$ Use a resource-allocation graph
Multiple instances of a resource type $\rightarrow$ Use the bankers algorithm

Questions from lecture

which 4 conditions most hold for a deadlock
- and describe them
- mutual exclusion
  - Two processes lock each other out due to interrupt
- hold and wait
  - Processes is waiting for resources to be released by other process
- circular wait
  - two or more processes are waiting for each other to release resources
- no pre-emption
  - cant remember

Describe the resource graph
- Graph the shows the dependency of resources and processes

What methods are there to handle deadlocks
- avoid deadlock
- allow deadlocks and recover
- ignore them

What is a safe state
- A state where deadlocks cant occur

Describe the general idea of bankers algorithm
- each process says how many resources they require, and the scheduler assigns the so no deadlocks occur

How can a deadlock be detached
- multiple processes are waiting for each other, and this can be detached with a resource graph

How can you recover from a deadlock
- kill processes that are in the deadlock until its resolved.

If you don't feel super comfy with deadlocks. talk about locks, and synchronization.

Question 6 - Memory Management Strategies - ch 8

notes

Introduction

Detailed description of various ways of organizing memory hardware
Various memory-management techniques, including paging and segmentation
To provide a detailed description of the Intel Pentium, which supports both pure segmentation and segmentation with paging

Keywords

1. *** Background *** - How do we access memory? - How/why do we need to enforce protection between processes? The base/limit registers to show a memory space. (p.347 book, p.353 pdf) - Address binding (a program assumes first address is 0000, but it rarely is). - Logical Addresses vs Physical Addresses (p.363 book, p.369 pdf)
2. *** Swapping *** (Moving processes between memory and disk).
3. *** Contigious Memory Allocation *** (One of three memory allocation schemes OS's can use).
4. *** Segmentation *** (Another one). First fit, Best fit, Worst fit.
5. *** Paging *** (Another one, often the one that is used due to eliminating external fragmentation completely).
6. *** Page table *** (Hierarchical paging, hashed page tables, and inverted page tables).
7. *** Pro / Cons *** (Paging doubles memory access time (without TLB(Translation lookaside buffer)), so why use it?).

Questions from lecture

In which stages can address binding happen.

compile time

If memory location known a priori, absolute code can be generated; must recompile code if starting location changes

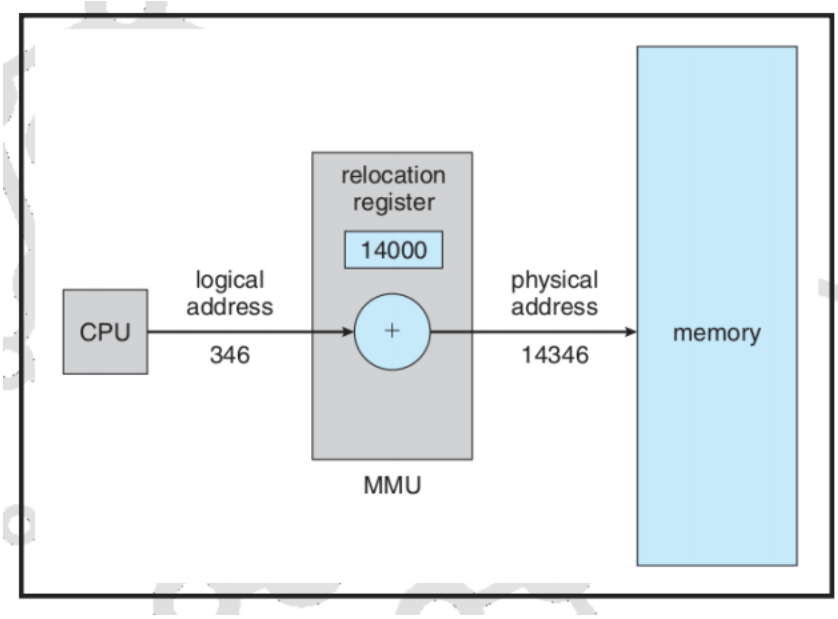
Load time

Must generate relocatable code if memory location is not known at compile time

Execution time

Binding delayed until run time if the process can be moved during its execution from one memory segment to another

What is an relocation register



offset

describe dynamic loading and dynamic linking

dynamic loading is when we load the libraries after load time

dynamic linking is when we set the offset after the program have been loaded

describe swapping and the back stores role

Swapping is when you store memory pages on cold storage due to low/no remaining memory (extremely slow)

back stores role

describe some algorithm for the dynamic storage-allocation problem

First Fit

Worst Fit

Best Fit

difference between external and internal fragmentation

Internal fragmentation is when there is unused memory within a process allocation

external is when there is unused memory between two processes

describe segmentation and paging.

segmentation is when you segmentation a program into data, execution code, etc

paging is when you split a program into pages, so there is no external fragmentation, but only internal on the last page of the program.

Question 7 - Virtual memory - ch 9

notes

Introduction

Virtual address space

Demand pages

Means we wont have to load all pages into memory, so processes can start faster. this uses a

page fault

Reference to a page that is not in memory, when this happens the OS check is if valid or not, if its not valid the os will throw an exception, and if its valid the page is loaded into memory, while this is happening the processes is put in a hold state(its a trap) and continued when the page is loaded into memory

hardware demand

page table with valid/invalid

performance of demand paging

Page replacement

first in first out
Least recently used
can use system clock or a stack with dual pointers, and move it to the top if its used. and always picks the one in button if its used.
Needs special hardware, its still slow, due to pointers needing to being changed
Second chance
clock, first in first out, with a reference bit
Least frequently used
Most frequently used
Optimal page replacement
Doesn't exist as we do not know the future.
But its good for comparing with other to see how they compare to the optimal.

Page allocation

Fixed allocation
equal allocation
dynamic, allocates based on size
priority allocation
shift memory from low priority to high priority,

Working set

Questions from lecture

Describe demand paging

Bring pages into memory when it is needed, meaning less I/O, less memory needed and faster response.

Describe copy on write

"Everyone" has a single shared copy of the same data until it's written, and then a copy is made

We fork the file so both the parent and the child are using the same file. when a write happen we copy the file so the thread that child and parent aren't working on the same file
needs rewriting.

Describe the page replacement algorithms

FIFO

First in first out. removed the oldest page in the buffer if requested one isn't there & and its out of space.

LRU

Least recently used.
Uses past knowledge.
Replaces page that has not been used in the most amount of time
Associate time of last use with each page

Counter implementation.

every page has a counter.

stack implementation

double link form.

Clock

circular clock

LFU

Least Frequently Used
replaces page with smallest count
Can give problems with some pages with lots of use gets "stuck" in the buffer.

MFU

based on the argument that the page with the smallest count was probably just brought in and has yet to be used

Optimal

Impossible as we don't know the future.

What is Beladys Anonmaly

Adding more frames can cause more page faults!

Describe trashing

If a process does not have enough pages, it gets a lot of page faults causing a lot of pages to be swapped out, only to be swapped in again.

Describe the working-set model.

The working set model, tries to predict how many frames we need to allocate for a process. at any given time, (it's dynamic)

Explain buddy system

Allocate memory from a fixed size segment.

Keywords

1. **Introduction / Benefits**
- Why do we need this?
- The relationship between virtual and physical memory.

2) **Demand paging**
- Concepts, Page-replacement algorithms, Page-faults
- Performance of demand paging.

3) **Copy-on-Write**
- General idea: why keep identical copies of pages in memory?

4) **Page replacement**
- If we run out of memory, which pages should we replace?)

FIFO
First in first out. removed the oldest page in the buffer if requested one isn't there & and its out of space.

LRU
Least recently used.
Uses past knowledge.
Replaces page that has not been used in the most amount of time
Associate time of last use with each page

Counter implementation.
every page has a counter.

stack implementation
double link form.

Clock
circular clock

LFU
Least Frequently Used
replaces page with smallest count
Can give problems with some pages with lots of use gets "stuck" in the buffer.

MFU
based on the argument that the page with the smallest count was probably just brought in and has yet to be used

Optimal
Impossible as we don't know the future.

5) **Allocation of frames**
- Which process get n number of frames

6) **Thrashing**
- Page-replacement gone wild, pdf p450

7) **Memory-Mapped Files**
- Hard to read

Question 8 - file system & Implementing filesystems - ch 10 and 11

May ask things about process control block

Notes

objectives

Explain the function and interfaces of file systems
Discuss file-system design tradeoffs, including access methods, file sharing, file locking, and directory structures Explore file-system protection
Describe the details of implementing local file systems and directory structures
Describe the implementation of remote file systems
Discuss block allocation and free-block algorithms and trade-offs

File control block

File permissions
File dates (created, access, edited)
add more

Questions from lecture

Describe different directory structures and their implementations single level One directory for all users
two level one Directory or each user tree structure infinite number of directories General graph Link only to files. ACYCLIC-GRAPH Link
Describe protection measures in file systems
Creator can set perms on a file.
Access control list Read Write, Execute Append Delete List
Describe the layers in a file system
Application programs. Doesnt care about anything logical file system. Cares about user rights file-organization module.
Basic file system. I/O control. Knows how to talk with the different devices. devices.
What does a device driver do?
Translator from file id, to sector on drive.
What does a FCB contain
File permissions Size Owner, group, ACL(Access control list) File dates, created, acess, edited File data blocks, or pointers to data blocks (INode)
Describe different methods for locating data for a file
INDEXED ALLOCATION
LINKED SCHEME
MULTILEVEL INDEX
COMBINED SCHEME
Describe methods for free space management
Free list Linked list

Keywords

Intro
i-node

Files
- Files operations
- How you read the files. linked, segmented, random.

Directories
- How do we know what files and subdirectories are in the directory.

Disk structure

File system mounting

File sharing

File access(Protection/permissions)
    Owner groups

Recovery

Network files

Allocation

Free Space management

Question 9 -Mass-Storage Structure and I/O Systems - ch 12 and 13

notes

Disk

1-dimensional array of logical blocks
Sector 0 is the first sector on the outermost track

Disk - Scheduling

SSTF Shortest seek time first. may cause starvation
Scan Go from lowest index, to highest, and back to lowest, also known as elevator algor.
C-scan Go from lowest index, to highest, and back to lowest without reading.
look Go from lowest index in queue, to highest in queue, and back to lowest in queue without reading.

Disk - Sector

Header information, Data, ecc code

Questions from lecture

What are the similarities and differences between NAS and SAN
TODO Describe the disc scheduling algorithms;
FCFS,
First come first serve
SSTF,
Shortest search first
SCAN,
all the way from first sector to last sector and back.
LOOK
Like scan by only goes from element to element, and doesn't start from the beginning.
C-Scan,
like scan but doesn't read on the way back
C-look Like scan but doesn't read on the way back
Describe sector sparing If the drive encounter a sector it can't write to it will then jump and write to a different sector that it has already allocated.
Describe raid and the mechanisms used raid 0 split the data on multiple disks for more speed raid 1

copy the data on multiple disk if one fails you wont loose the data. raid 2 (Bits striped)
two groups of disk, one group is data, one disk stores the correction code. raid 3(bytes striped)
dedicated storage disks and a dedicated parity disk
raid 4(blocks Striped) same just with blocks
raid 5(blocks striped, and distributed parities)
raid 6(blocks striped, and two distributed parities)
raid 10
What is stable storage and how can yo achieve it
Name some of the characteristics of i/o devices
Explain caching, spooling and device reservation

Keywords

Introduction **Ex**
Disk Scheduling **FCFS**
Look
SSFT
FCFS
Disk manegment •
Raid (read) http://igoro.com/archive/how-raid-6-dual-parity-calculation-works/ **Raid 5/6**
raid 0 split the data on multiple disks for more speed raid 1
copy the data on multiple disk if one fails you wont loose the data. raid 2 (Bits striped)
two groups of disk, one group is data, one disk stores the correction code. raid 3(bytes striped)
dedicated storage disks and a dedicated parity disk
raid 4(blocks Striped) same just with blocks
raid 5(blocks striped, and distributed parities)
raid 6(blocks striped, and two distributed parities) Parity is often calculated but xoring the data and saving the output as a parity.
raid 10 I/O devices buses, and controllers and port
I/O Hardware How do to I/O talk to hardware Busy bit where the host waits until the devices is ready. Interrupts blocks the I/O devices Direct memory access.
I/O Streams

Question 10 - System Protection - ch 14

Keywords

Introduction
Domains
access matrix
Revocation of access rights.
capability based system
Need to know
Language based protection
Types of attacks

Questions from lecture

Describe how an access matrix is used and what it contains
How can you implement the access matrix
What security measure levels should you handle
Describe the principles in the following attacks: stack and buffer overflow, virus, trap door.
Describe differences between symmetric and asymmetric cryptology
What is an authenticator

Question 11 - System Security - ch15

Keywords

Introduction
Types of attack
Trojan horse
logic bomb
symmetrical and asymmetrical encryption
1 key vs 2 key
b64 vs rsa
RSA
User authentication
How to store passwords(not everyone should have access).
Firewalls

Questions from lecture

Question 12 - Virtual Machines cp-16

Keywords

1) **Introduction** What is it. Containers,
Why should we use it? Security (Explain what virtualization is, and what a virtual machine is. The figure from above is good.)
2) **History** (This should go with the introduction if mentioned)
3) **Benefits and Features** Security Safer as machines are isolated from each other. Testing software in an isolated environment. live Migration We can move machines to other hardware without stopping them.
Snapshots
there's a tiny bit of performance overhead.
4) **Building Blocks** (Very important - explains how it works, two method especially: Trap-and-Emulate/Binary translation, See p717)
Trap and emulate , block systems calls and emulated what they did. Binary translation, translates it to something that doesn't cause it to damage the host, or if the hardware doesn't support the function.
5) **Types of Virtual Machines and Their Implementations** Type 0 hyper visor - Hardware based. Type 1 Hypervisor - Specific vm based os vmware(window uses this for hyper-v) type 2 Hypervisor - Normal os with a vm container program
Paravirtualization
6) **Virtualization and Operating-System Components** Nested page tables. (Extends (4) and goes into details of how many of the concepts we learned, are transferred into the virtual system.)

Questions from lecture

Describe the role of the VMM/hypervisor:
The mange the virtual machines, hardware access, and port forwarding, and such, may only do virtual switches, and more.
What is an emulator
Program that emulates a different operating system or set of hardware.
Name some of the benefits of virtualisation
the processes are separated within the same hardware, and you can run multiple different different operating systems on the same hardware.
Snapshots
use hardware more efficiently
Live migration
Describe the "trap and emulate" and "binary translation"
We don't want the user to access the kernel mode on the hardware.
So the VMM does all the kernal operations, its just very expensive.

Why are nested page tables used
Because each VM has its own memory.
How is live-migration preformed
1. create new vm on new host
2. move read only pages to new host
3. move read/write pages to new host
4. move dirty pages to new host, and check if there are changes to read/write pages and move those.
5. pause the vm, move the last pages, and point network traffic to new host.
6. shut down old vm.

Question 13 - Networks and Distributed Systems cp-17

Keywords

1) **Introduction** - Advantages of Distributed Systems, such as Resource sharing/Speedup/Reliability. - Disadvantages would be a nice touch too. 2) **Types of Network-based Operating Systems** - Network OS/Distributed OS - What are the differences? 3) **Network Structure** - LAN and WAN, also known as Local Area Network and Wide Area Network, difference being how they are physically placed - WAN is the network between the different campus' here at SDU 4) **Communication Structure** - The gritty details of how the CPUs communicate, very important. - Especially stuff such as routing strategies. 5) **Communication Protocols** - Same as above 6) **Robustness** - How can we make the system more robust regarding system failure, or network failure 7) **Design Issues** - What are some typical design issues that needs to be handled, **notice how this section could let many here learn what a typical design section might be** 8) **Distributed File Systems** - OpenAFS, NFS 9) **Examples** Book offers several examples, mentioning these can often be a nice touch, when explaining the rest.