370 notes  
  
File Locking  
Provided by some operating systems and file systems  
 -similar to reader-writer locks

Shared lock similar to reader lock – several processes can acquire concurrently  
Exclusive lock similar to writer lock

File Structure  
None – sequence of words, bytes  
Simple record structure  
 Lines  
 Fixed Length  
 Variable length  
Complex structures  
-formatted document  
-relocatable load file  
  
can simulate the last two with the first method by inserting appropriate control characters  
  
who decides?  
Operating System  
Program

Access Methods  
Sequential Access vs Direct Access  
  
Disk Structure  
Disk can be subdivided into partitions  
can be RAID protected against failure  
disk or partition can be used raw – without a file system or formatted with a file system  
partitions also known as minidisks, slices  
Entity containing file system known as a volume

Operations performed on a directory  
  
Search for a file  
Create a file  
Delete a file  
List a directory  
Rename a file  
traverse the file system  
  
Single level vs two level directory vs tree structure directory

Operating System Services  
provides an environment for execution of programs and services for programs and users  
One set of OS services provide functions that are helpful to the user  
- User interface (Almost all have a UI varies between command line, graphics User interface, and Batch)  
- Program Execution (The system must be able to load a program into memory and to run that program, end execution, either normally or abnormally (indicating error))  
- I/O operations (A running program may require I/O  
- File System Manipulation  
- Communications  
- Error detection  
  
another set of OS functions exists for ensuring efficient operation of the system via resource sharing  
- Resource allocation  
-Accounting  
- Protection and security

Register m-fastest and most accessible memory, very limited resource

Types of system calls  
-Process control (create and terminate processes, end and abort, load ans execute, get and set attributes, wait for time, allocate and free memory, dump memory if error, debugger for determining bugs, locks for managing access to shared data between proceses)  
- File management (create, delete, open, close, read, write)  
- Device Management (request device, release device, read, write, reposition, get device attributes, set device attributes, logically attach or detach)  
- Information maintenance (get time set time, get system data, set system data, get and set file or device attributes)  
- Communications (  
- Protections (control access to resources, get and set access)

System Programs  
Provide environment for development and execution  
-File manipulations  
-status info  
-file info  
-file modification  
-programming language support  
-Program loading and execution  
-communications  
-background services  
-application programs

System Programs  
Provide a convenient environment for program development and execution  
**File Management** -create, delete, copy, rename, print, dump, list, and generally manipulate files and directories  
**Status information** -Some ask system for info -date time amount of available memory, disk space, number of users  
- Others provide detailed performance, logging, and debugging information  
- Typically these programs format and print the output to the terminal or other output  
- Some systems implement a registry -used to store and retrieve configuration information

**File Modification**  
- Text editors to create and modify files  
- Special commands to search contents of files or perform transformations of the text

**Programming-language support**  
- compilers, assemblers, debuggers, and interpreters sometimes provided.

**Program loading and execution**  
absolute loaders, relocatable loaders, linkage editors, overlay loaders, debugging systems for higher level and machine language

**Communications**  
Provide mechanism for creating virtual connections among processes users and computer systems  
- send messages to one another’s screens  
-browse web pages  
- send emails  
-log in remotely

**Linkers and Loaders**  
-Sources files compiled into object files known as a relocatable object file  
-Linker combines relocatable object files into binary executable file

Forking creates a child process executes the same code from the same point being forked, parent returns process id of the child from the fork, the child returns a zero

**Chapter 3**

Chart, box and whisker chart

Description automatically generatedBatch systems -jobs  
time shared systems -user programs or tasks

Job and process used interchangeably

**Process Concept**  
Multiple parts  
program code aka text  
program counter and process registers (execute machine code commands)  
stack contains temporary data  
 Function parameters, return addresses, local variables  
Data section containing global variables  
Heap contains dynamically allocated memory

Another Example  
Diagram

Description automatically generated

Program becomes a process when executable file is loaded into memory

One program can become several processes because multiple users can run a single program

Diagram

Description automatically generated**Process State  
new**: process being created  
**running**: instructions being executed  
**waiting**: the process is waiting for some event to occur  
**ready**: the process is waiting to be assigned a processor  
**terminated**: the process has finished execution

**Process control block**  
Information stored about an active process  
**Process State** -running, waiting etc  
Program counter -location of instruction to execute  
CPU registers -contents for process centric data  
CPU Scheduling priority and queue  
Memory Management info – memory allocated to the process  
Accounting information -CPU used, clock time elapsed, time limits  
IO Status -IO devices allocated to process list of open files

**Process Scheduling**objective of multiprogramming process running at all times to maximize CPU utilization  
objective of time sharing to switch a CPU core among processes  
Process scheduler selects process for execution on a core  
 each CPU core can run 1 process at a time  
 System with single CPU core

Maximize CPU use, quickly switch process onto CPU for time sharing  
Process Scheduler selects among available processes for next execution on CPU  
Maintains scheduling queues of processes  
 Job queue -set of all processes in the system (haven’t been moved into ready queue)  
 Ready Queue -set of all processes in main memory, ready and waiting to execute

**Threads**  
So far process has a single thread of execution   
Consider having multiple program counters per process  
multiple locations executing at once  
Must have multiple storage for thread details multiple counters in PCB

**Scheduling Queues -ready queue**  
As process enter system they are put into a ready queue  
 waiting to execute on CPU’s core  
 generally stored as a linked list

**Wait queue**  
once process allocated CPU core executes for a while, either terminates, interrupted or waits for an event  
Suppose process makes an IO request to a device, device run significantly slower than processors  
Process will wait for IO device to become available and placed in wait queue

Schedulers  
Short term scheduler  
selects which process shoyld be executed next

Long term scheduler  
Selects which process should be brought to the ready queue

Process can be described as: IO bound or CPU bound

Addition of Medium term scheduling  
Medium term scheduler can be added if degree of multiple programming needs to decrease  
 remove process form memory, store on disk  
 bring back in from disk to continue execution: swapping

Multitasking in mobile systems  
typically allow one process to run  
 due to power, memory, and interface (like screen real estate)  
allows for multiple background processes

Context Switch  
When CPU switches to another process syetm must  
save state of old process  
load saved state for new process via a context switch  
Context of a process represented in the PCB  
Context switch time is overhead  
 system does non useful work while switching  
 The more complex the OS and the PCB  
 the longer the context switch

Time dependent on hardware support  
 Some hardware has multiple sets

**Process Termination**

Some OS do not allow child to exist if aprent has been terminated,  
  
 if a parent has been terminated it will result in cascading termination including all children grandchildren etc.

If no parent waiting (did not invoke wait()) process is a zombie  
  
If parent terminated without generating wait process is an orphan

**Interprocess Communications**

Message Passing vs Shared memory

Message Passing  
If two process iwsh to communicate they need to establish a communication link between them  
Excahnge messages via send/receive

Implementation issues  
How are the links established  
Can a link be associated with more than two processes  
How many links can there be between every pair communicating processes

Implementation of comm link  
Physical  
Shared memory  
Hardware bus  
Network  
  
Logical  
Direct or indirect  
Synchronous or asynchronous  
Automatic or explicit buffering

**Direct communication**

Processes must name each other explicitly  
Send(P, message)  
Receive(Q, message)  
  
Properties of comm link  
links established automatically  
A link is with one pair  
Between each pair there exists only one link

**Indirect communication**

Messages are directed and received from mailboxes aka ports  
each mailbox has a unique ID  
Processes can communicate if they share a mailbox

Properties of comm link  
Link established id process

Operations  
Create a new mailbox  
Send and receive messages through mailbox  
Destroy a mailbox

Primative defined as …

Mailbox sharing  
p1 p2 p3 share mailbox A  
p1 sends, p2 and p3 receive  
who gets the message  
SOLUTIONS  
allow a link to be associated with at most two processes OR  
allow only one process at a time to execute a receive operation OR  
allow system to select arbitrary receiver

Socket: an endpoint for communication  
 concatenation of IP address and port  
  
EX) the socket 161.25.19.8:1625 refers to port 1625 on host 161.25.19.8

Communication exists between a pair of sockets  
All ports below 1024 are well kown used for standard services

Special IP address 127.0.0.1 (loopback) to refer to system on which process is running (localhost)

Pipes  
Acts as a conduit allowing twon processes to communicate  
Issues:  
Is communication unidirectional or bidirectional?  
In case of two way communication is it half or full-duplex (half is walkie talkie, both can talk but one at a time, telephone call is full duplex as you can talk over one another)  
Must there exist a relationship (ie parent child) between comm processes

Ordinary pipes cannot be accesses from outside process that created it  
typically a parent process creates a pipe and uses it to comm with a child process that is created

**Chapter 4: Threads and concurrency:**multithreaded server architecture:  
Diagram

Description automatically generated

Benefits  
Responsiveness  
Resource Sharing

Challenges of multicore programming  
Dividing activities  
Balance  
Data splitting

Data dependency (tasks must sync)  
Testing and debugging

Parallelism -implies can perform more than 1 task simultaneously  
Concurrency -supports more than one task making progress  
Text, table

Description automatically generated

Data Parallelism  
distributes subsets of same data across multiple cores, same operation on each  
  
Task Parallelism  
distributes threads across cores, each thread performing unique operation

Amdahl’s law  
identifies performance gains from adding additional cores to an application has that has serial and parallel components  
S = serial portion  
N = processing cores

If 75% is parallel and 25% is serial moving from 1 to 2 cores results in 1.6 times speedup

Relationship between user threads and kernel threads  
-Many to one  
-One to one  
-many to many

Midterm Feb 24  
Open Notes

Java threads managed by jvm  
implemented using underlying OS  
Created by extending thead class or implementing runnable interface.

Implicit Threading  
Creation and management of threads done by compilers and runtime libraries rather than programmers

Thread Pools

OpenMP

Thread cancellation  
terminate a thread before its finished

Thread local storage (TLS) allows each thread to have its won data  
Useful when you do not have control over thread creation process (thread pools)  
Different from local variables  
Similar to static data  
 TLS is unique in each thread

Operating System examples

Windows threads  
one to one mapping kernel level  
data structures  
 ethread (executive thread block)  
 kthread (kernel thread block)  
 TEB (thread environment block)

Linux Threads  
refers to them as tasks rather than threads  
thread creation done through clone() system call  
clone() allows a child task to share the address space of the parent task (process)  
flags control behavior  
struct task\_struct points to process data structures (shared or unique)  
new task is created when clone() system call is made

* rather than copying all data structures, new task points to data structures of parent task depending on set of flags passed to clone()

CPU Scheduling  
Program 3 is assigned  
may have a lecture during lab

Short term scheduler handles the ready queue

Round Robin   
each process gets a small amount of CPU time  
after time has elapsed process is preempted and added to end of ready queue  
main idea: passes cpu time among processes

Turnaround time varies with time quantum

**BE SURE TO PRACTICE TURNAROUND TIME CALCULATIONS THEY WILL BE ON THE EXAM**

Multi-level queue  
separate ques   
foreground and background  
each has its OWN scheduling algorithm   
foreground often uses RR while background tends to use FCFS (first come first serve)  
  
Multilevel feedback queue  
processes can move between the queues  
number of queues  
scheduling alg for each queue  
method used to determine when to upgrade a process and when to demote a process

Starvation: processes not getting CPU time and aren’t executing (never gets put in que)

RR: round robin  
FCFS: first come first serve  
SJF: shortest job first  
  
RR has better response but worse turnaround time compared to SJF

Priority based scheduling  
for real time scheduling, scheduler must support preemptive priority based scheduling  
for hard real time must provide ability to meet deadlines

Processes have new characteristics: periodic ones require CPU at constant intervals  
Has processing time t, deadline d, period p  
0 <= t <= d <= p

Monotonic scheduling  
smaller periods are prioritized, longer periods have less priority

Chapter 6: synchronization tools

Each semaphore has a associated waiting queue

Monitors  
incroeect use of semaphore operations  
- singal (mutex) … wait (mutex) (example of incorrect operation)  
a high level abstraction that provides convenient and effective mechanism for process synchronization

Chapter 9 main memory

Description of memory organization hardware and memory management techniques  
  
background

Memory Protection  
protection bit associated with each frame to indicate read only or read write access  
Valid invalid bit attached to each entry in the page table

Can use PTLR page table length register  
any violations result in trap kernel

Table

Description automatically generated  
shared pages  
- shared code (one copy of read only code shared among processes)  
- similar to threads sharing process space, also useful for interprocess communication  
Private code and data  
- each process keeps a separate copy

Graphical user interface, text, application

Description automatically generated

disk scheduling  
looking at how the OS can take advantage of scheduling reads and writes to minimize the delay

Networks and Distributed Systems  
A distributed system is a collection of loosely coupled nodes interconnected by a communications network

Nodes variously called processors computers machines hosts.

Site is a location of a machine  
Node refers to a specific system  
Generally a server has a resource a client node at a different site wants to use

Nodes may exists in peer to peer, hybrid, or client server configs  
  
LAN designed to cover small geographical area  
consists of multiple computers, peripherals, and routers providing access to other networks.

Distributed Operating Systems  
  
Users not aware of multiplicity of machines  
Access to remote resources similar to access to local resources  
  
Data Migration transer data by  
- transferring entire file  
- transferring only portions of file necessary for immediate task

Computation Migration -transfer computation, rather than the data, across the system  
-via remote procedure calls  
- via messaging system

Distributed Operating Systems (Cont)  
Process Migration – execute an entire process, or parts of it, at different sites  
- load balancing -distribute processes across the network to even the workload  
- Computation speedup  
- Hardware Preference  
- Software preference  
- Data Access

Design Issues of Distributed Systems  
-Robustness  
-Transparency (can distributes system be transparent to the user both in terms of where files are stored and user mobility)  
- Scalability

Robustness  
hardware failures can include failure of a link failure of a site, and loss of a message  
- A fault tolerant system can tolerate a certain level of failure  
- Involves  
1. Failure detection  
2. Reconfiguration  
3. Recovery