CHAPTER 2 TERMS

batch processing - Pertaining to the technique of executing a set of computer programs such that each is completed before the next program of the set is started.
batch system - The central idea behind the simple batch-processing scheme is the use of a
piece of software known as the monitor . With this type of OS, the user no longer has direct access to the processor. Instead, the user submits the job on cards or tape to a computer operator, who batches the jobs together sequentially and places the entire batch on an input device, for use by the monitor. Each program is constructed to branch back to the monitor when it completes processing, at which point the monitor automatically begins loading the next program.
execution context - or *process state*, is the internal data by which the OS is able to supervise and control the process. This internal information is separated from the process, because the OS has information not permitted to the process. The context includes all of the information that the OS needs to manage the process and that the processor needs to execute the process properly
interrupt - Early computer models did not have this capability. This feature gives the OS more flexibility in relinquishing control to and regaining control from user programs.
job - A single program
job control language - a special type of programming language used to provide instructions to the monitor
kernel - or *nucleus*, which contains the most frequently used functions in the OS and, at a given time, other portions of the OS currently in use
memory management - The OS has five principal storage management responsibilities: Process isolation, Automatic allocation and management, Support of modular programming, Protection and access control, Long-term storage *microkernel* - A small privileged operating system core that provides process scheduling, memory management, and communication services and relies on other
processes to perform some of the functions traditionally associated with the operating system kernel.
monitor - A programming language construct that encapsulates variables, access procedures, and initialization code within an abstract data type. The monitor's variable may only be accessed via its access procedures and only one process may be actively accessing the monitor at any one time. The access procedures are critical sections. A monitor may have a queue of processes that are waiting to access it.
monolithic kernel - A large kernel containing virtually the complete operating system, including scheduling, file system, device drivers, and memory management. All the functional components of the kernel have access to all its internal data structures and routines. Typically, a monolithic kernel is implemented as a single process, with all elements sharing the same address space.
multiprogrammed batch system - Processor is often idle, even with automatic job sequencing, I/O devices are slow compared to processor, objective: maximize processor use, job control language provided with job
multiprogramming - A mode of operation that provides for the interleaved execution of two or more computer programs by a single processor. The same as multitasking, using different terminology.
multitasking - A mode of operation that provides for the concurrent performance or interleaved execution of two or more computer tasks. The same as multiprogramming, using different terminology.
multithreading - is a technique in which a process, executing an application, is divided into threads that can run concurrently
operating system - a program that controls the execution of application programs and acts as
an interface between applications and the computer hardware. It can be thought of as having three objectives: Convenience, Efficiency, Ability to evolve
physical address - The absolute location of a unit of data in memory (e.g., word or byte in main memory, block on secondary memory).
privileged instruction - An instruction that can be executed only in a specific mode, usually by a supervisory program.
process - A program in execution. A process is controlled and scheduled by the operating system. Same as *task*.
Resident monitor – portion of monitor that is in the main memory and always available for execution.
real address - A physical address in main memory.
round robin - A scheduling algorithm in which processes are activated in a fixed cyclic order; that is, all processes are in a circular queue. A process that cannot proceed because it is waiting for some event (e.g., termination of a child process or an input/output operation) returns control to the scheduler.
scheduling - To select jobs or tasks that are to be dispatched. In some operating systems, other units of work, such as input/output operations, may also be scheduled.
serial processing - With the earliest computers, from the late 1940s to the mid- 1950s, the programmer interacted directly with the computer hardware; there was no OS. These computers
were run from a console consisting of display lights, toggle switches, some form of input device, and a printer. Two main problems: Scheduling - Most installations used a hardcopy sign-up sheet to reserve computer, Setup time - A single program, called a job , could involve loading the compiler
symmetric multiprocessing - A form of multiprocessing that allows the operating system to execute on any available processor or on several available processors simultaneously.
thread - A dispatchable unit of work. It includes a processor context (which includes the program counter and stack pointer) and its own data area for a stack (to enable subroutine branching). A thread executes sequentially and is interruptible so that the processor can turn to another thread. A process may consist of multiple threads.
time sharing - The concurrent use of a device by a number of users.
time-sharing system - Can be used to handle multiple interactive jobs, Processor time is shared among multiple users, Multiple users simultaneously access the system through terminals, with the OS interleaving the execution of each user program in a short burst or quantum of computation, objective: minimize response time, commands entered at terminal
virtual address – The address of store location in virtual memory.
uniprogramming - single-program systems

MISC DATA

Symmetric Multiprocesors – two or more similar processors of comparable capability. Processors share the same main memory and are interconnected by a bus or other internal connection scheme. Processors share access to I/O devices. All processors can perform the same functions. System is controlled by an integrated operating system that provides interaction between processors and their programs at the job, task, file, and data element levels. *ADVANTAGES* – Better performance if work can be done in parallel. Failure of a single processor doesn't halt machine. Incremental growth, adding a processor can enhance performance. Scaling, vendors can offer a range of products with different price and performance characteristics. *Multicore Computer* – combines two or more processors (cores) on a single piece of silicon (die). Each core consists of all the components of an independent processor. Include L2 and sometimes L3 cache. *Paging* – Allows processes to be comprised of several fixed size blocks. Program references a word by use of a virtual address (Page number, and an offset within the page). Each page may be located anywhere in memory. Allows dynamic mapping between the virtual address used in the program, and a real (physical) address in main memory.
Hypothetical Computer System:

IR = Opcode + Mem Addr IR = 3 Hex Digits
PC = 2 Hex Digits (equivalent to a Mem Addr) Data = IR(unsigned) = Mem Position
# of Opcodes = 2^n where n = # of bits in opcode
MemSize in bits = 2^n * (# of bits in IR) i.e n = # of bits in Mem Addr Mem Range = Mem Addr Size (2 Hex) = 00 – FF
Data Range = IR size (3 Hex) = 000 – FFF 1 Hex Digit = 4 bits \ \ 1 Octal Digit = 3 bits
Access Time Formula [MR = Miss Ratio] [HR = Hit Ratio = 1 – MR]

AvgTime = TL_1 * HR + (TL_1 + TL_2) * MR
SMP(Symmetric Multiprocessing) – schedules processes or threads across all of the processors. Several processes can run in parallel. Multiple processors are transparent to the user. OS takes care of scheduling of threads or processes on individual processors and of synchronization among processors.
Benefits of Threads vs Processes - Less time to create a new thread. Less time to terminate. Switching between threads takes less time. Enhance efficiency in communication between programs.
Process Elements Program Code – may be shared with other processes that are executing the same program. *Set of Data associated with that code. Process Control Block (includes the following)* - *Identifier* – uniquely associated with the process for identification. *State* – if it is executing it is in the running state. *Priority* – priority level relative to other processes. *Program counter* – address of the next instruction in the program to be executed. *Memory pointers* – pointers to the program code and data associated with this process, plus any memory blocks shared with other processes. *Context Data* – data that are present in registers in the processor while the process is executing. *I/O Status info* – outstanding I/O requests, I/O devices assigned to this process, a list of files in use by the process.
Accounting Info – may include the amount of processor time and clock time used, time limits, account numbers.
Kernel-level threads - A thread must use the scheduler provided by the OS.
Ready processes - Processes that reside in main memory and can be selected by the scheduler.
Monolithic kernel - Includes virtually all the OS functionality in one large block of code.
Division by zero - An example of a program interrupt.
Supervisor mode - Mode used by kernel to run its functions.
LRU - Effective strategy is to replace a block that has been in the cache the longest with no references.

CHAPTER 1 TERMS

cache memory - A memory that is smaller and faster than main memory and that is interposed between the processor and main memory. The cache acts as a buffer for recently used memory locations.
direct memory access - used when large volumes of data are to be moved, performed by a separate module on the system bus or incorporated into an I/O module, Transfers the entire block of data directly to and from memory without going through the processor, processor is involved only at the beginning and end of the transfer, processor executes more slowly during a transfer when processor access to the bus is required, More efficient than interrupt-driven or programmed I/O.
hit ratio - defined as the fraction of all memory accesses that are found in the faster memory
input/output - Move data between the computer and its external environment. The external environment consists of a variety of devices, including secondary memory devices (e.g., disks), communications equipment, and terminals.
instruction -
instruction cycle - The processing required for a single instruction, the two steps are referred to as the fetch stage and the execute stage. Program execution halts only if the processor is turned off, some sort of unrecoverable error occurs, or a program instruction that halts the processor is encountered.
instruction register - The fetched instruction is loaded into the instruction register (IR). The instruction contains bits that specify the action the processor is to take. The
processor interprets the instruction and performs the required action. In general, these actions fall into four categories: Processor-memory, Processor-I/O, Data processing, Control
Interrupt - Interrupt the normal sequencing of the processor, provided to improve processor utilization, most I/O devices are slower than the processor, processor must pause to wait for device, wasteful use of the processor
interrupt-driven I/O - is for the processor to issue an I/O command to a module and then go on to do some other useful work. The I/O module will then interrupt the processor to request service when it is ready to exchange data with the processor. The processor then executes the data transfer, as before, and then resumes its former processing. (More efficient that programmed I/O) Drawbacks: 1. The I/O transfer rate is limited by the speed with which the processor can test and service a device. 2. The processor is tied up in managing an I/O transfer; several instructions must be executed for each I/O transfer.
I/O module - Move data between the computer and its external environment. The external environment consists of a variety of devices, including secondary memory devices (e.g., disks), communications equipment, and terminals.
locality - Memory references by the processor tend to cluster, Data is organized so that the percentage of accesses to each successively lower level is substantially less than that of the level above, can be applied across more than two levels of memory
main memory - Stores data and programs. This memory is typically volatile; that is, when the computer is shut down, the contents of the memory are lost
multicore - also known as a chip multiprocessor, combines two or more processors (called cores) on a single piece of silicon (called a die)
processor - Controls the operation of the computer and performs its data processing (CPU)
program counter - holds the address of the next instruction to be fetched
programmed I/O - the I/O module performs the requested action and then sets the appropriate bits in the I/O status register but takes no further action to alert the processor. It does not interrupt the processor. Thus, after the I/O instruction is invoked, the processor must take some active role in determining when the I/O instruction is completed. For this purpose, the processor periodically checks the status of the I/O module until it finds that the operation is complete. With programmed I/O, the processor must wait a long time for the I/O module of concern to be ready for either reception or transmission of more data. The processor, while waiting, must repeatedly interrogate the status of the I/O module. As a result, the performance level of the entire system is severely degraded.
secondary memory - External, nonvolatile memory
spatial locality - refers to the tendency of execution to involve several memory locations that are clustered
system bus - Provides for communication among processors, main memory, and I/O modules.
temporal locality - refers to the tendency for a processor to access memory locations that have been used recently

```
// getpid() gives the process
// id and getppid() gives the
// parent id of that process.
```

CHAPTER 3 TERMS

blocked state - A process that cannot execute until some event occurs, such as the completion of an I/O operation.
child process - When one process spawns another, the former is referred to as the parent process, and the spawned process is referred to as the child process
exit state - A process that has been released from the pool of executable processes by the OS, either because it halted or because it aborted for some reason
interrupt -
kernel mode - A privileged mode of execution reserved for the kernel of the operating system. Typically, kernel mode allows access to regions of main memory that are unavailable to processes executing in a less-privileged mode and enables execution of certain machine instructions that are restricted to the kernel mode. Also referred to as system mode or privileged mode .
mode switch - A hardware operation that occurs that causes the processor to execute in a different mode (kernel or process). When the mode switches from process to kernel, the program counter, processor status word, and other registers are saved. When the mode switches from kernel to process, this information is restored.
new state - A process that has just been created but has not yet been admitted to the pool of executable processes by the OS. Typically, a new process has not yet been loaded into main memory, although its process control block has been created.
parent process – a process that spawns another
Preempt - Reclaiming a resource from a process before the process has finished using it.
privileged mode - Same as kernel mode.
process control block – The manifestation of a process in an operating system. It is a data structure containing information about the characteristics and state of the process.
process image - All the ingredients of a process, including program, data, stack, and process control block.
process switch - An operation that switches the processor from one process to another, by saving all the process control block, registers, and other information for the first and replacing them with the process information for the second.
program status word(PSW) - A register or set of registers that contains condition codes, execution mode, and other status information that reflects the state of a process.
ready state - A process that is prepared to execute when given the opportunity.
round robin - each process in the queue is given a certain amount of time, in turn, to execute and then returned to the queue, unless blocked
running state - The process that is currently being executed. For this chapter, we will assume a computer with a single processor, so at most one process at a time can be in this state.
suspend state - When all the processes in main memory are in the Blocked state, the OS can suspend one process by putting it in the Suspend state and transferring it to disk. The space that is freed in main memory can then be used to bring in another process.
swapping - A process that interchanges the contents of an area of main storage with the contents of an area in secondary memory
system mode - Same as kernel mode .
trace - A sequence of instructions that are executed when a process is running
trap - An unprogrammed conditional jump to a specified address that is automatically activated by hardware; the location from which the jump was made is recorded.
user mode - The least-privileged mode of execution. Certain regions of main memory and certain machine instructions cannot be used in this mode.

PID - An element of the process control block.
pthread library - A user-level thread implementation.
Multiple Interrupts: Sequential - Disable interrupts while an interrupt is being processed.
if ((pid=fork()) == 0) - Child process code.
a.What is the goal of the suspend state in the five-state process model with one suspend state?
To free a space in main memory and bring in another process
b.What is the main objective of a time-sharing system?
Minimize response time of processes
c.Describe why in a pure ULT strategy, a multithreaded application cannot take advantage of multiprocessing.
The single-threaded process at the kernel level that represents the multithreaded processes at the user level can only run in one CPU at a particular time
d.What is the relationship between the cache memory and the block size?
Big block size = small number of records in the cache memory (replacement probability increases.
Small block size = destroying the principle of locality.
e.What is the major disadvantage of a system call?
Change from user mode to supervised mode = context switch.
f.What is the major advantage of DMA against interrupt-driven I/O?
DMA is performed by a separate module on the system bus or incorporated into an I/O module, therefore the processor is not interrupted during word transfer. The CPU is only used at the beginning and end of the transfer.

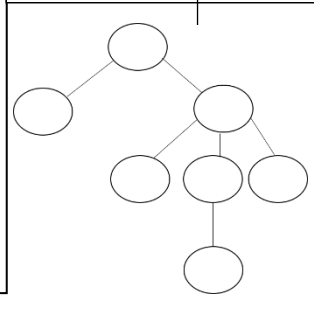
```
#include <pthread.h>
#include <stdio.h>
#define NTHREADS 5
#define COUNT 5
void *inc_x(void *x_void_ptr)
{
   int *x_ptr = (int *)x_void_ptr;
   int i;
   for (i=0; i< COUNT; i++)
   (*x_ptr)++;
   return NULL;
}
int main()
{
   static int x = 0;
   pthread_t tid[NTHREADS];
   int i;
   printf("Initial value of x: %d\n", x);
   for(i=0;i<NTHREADS;i++)
   {
   /*if(false) doesn't execute -> pthread_create on success returns 0 = false
   **else returns true and executes the print statements.
   **int pthread_create(pthread_t *thread, const pthread_attr_t *attr,
   **   void *(*start_routine) (void *), void *arg); */
   if(pthread_create(&tid[i], NULL, inc_x, &x))
      {
         fprintf(stderr, "Error creating thread\n");
         return 1;
      }
   }
   // Wait for the other threads to finish - > pthread_join()
   for (i = 0; i < NTHREADS; i++)
         pthread_join(tid[i], NULL);
   printf("Final value of x: %d\n",x);
   return 0;
}
```

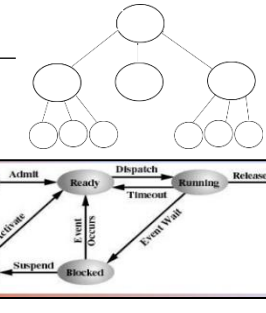
```
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>

int main()
{
   pid_t pid;
   int x = 0;
   int i, j, k, l;
   for (i = 0; i < 2; i++)
      if ((pid = fork()) == 0)
         break;
   if (pid == 0)
   {
      if (i == 1)
      {
         for (j = 0; j < 3; j++)
            if ((pid = fork()) == 0)
               break;
         if (pid == 0)
         {
            if (j == 1)
               if ((pid = fork()) != 0)
                  wait(0);
         }
         else
         {
            for (k = 0; k < 3; k++)
               wait(0);
         }
      }
   }
   else
   {
      for (l = 0; l < 2; l++)
         wait(0);
   }
   printf("I am the process %d\n", getpid());
   sleep(10);
   return 0;
}
```

```
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>

int main()
{
   pid_t pid;
   int i,j;
   for (i =0; i <3; i++)
      if((pid=fork())==0)     // child process
         break;
   if (pid == 0)
   {
      if (i%2 == 0)
      {
         for (j =0; j <3; j++)
               if((pid=fork())==0)     // child process
                     break;
      }
   }
   else   // parent process
      for (int i=0;i<3;i++)
         wait(NULL);
   sleep(10);
   return 0;
}
```





CHAPTER 4 TERMS

kernel-level thread - Thread management is done by the kernel; no thread management is done by the application; Windows is an example of this approach. Advantages: The kernel can simultaneously schedule multiple threads from the same process on multiple processors, if one thread in a process is blocked, the kernel can schedule another thread of the same process, Kernel routines can be multithreaded. Disadvantages: The transfer of control from one thread to another within the same process requires a mode switch to the kernel
lightweight process – see thread
message - A block of information that may be exchanged between processes as a means of communication.
multithreading - The ability of an OS to support multiple, concurrent paths of execution within a single process
port -
process - the unit of resource allocation and a unit of protection, A virtual address space that holds the process image, Protected access to processors, other processes (for interprocess communication), files, and I/O resources (devices and channels) *task* – see task
thread - The unit of dispatching
user-level thread - All thread management is done by the application; the kernel is not aware of the existence of threads. Advantages: Thread switching does not require kernel mode privileges, Scheduling can be application specific, ULTs can run on any OS. Disadvantages: In a typical OS many system calls are blocking as a result, when a ULT executes a system call, not only is that thread blocked, but all the threads within the process are blocked, in a pure ULT strategy, a multithreaded application cannot take advantage of multiprocessing, The single-threaded process at the kernel level that represents the multithreaded processes at the user level can only run in one CPU at a particular time.

What is the major disadvantage of not having privileged instructions? A. Unrestricted user access to the data on the hard drive. **In a vectorized interrupt system, an interrupt can only be interrupted by an interrupt of higher priority**. True. **A program in execution is called**. A process. **What is the major reason for the success of modular kernels?** They let users add extensions to the kernel. **Each process has its own Process Control. Block.** True. **Which of these events can move a process from the running state to the blocked state?** The process performs a system call. **Which of the following statements apply to the program.cs.uh.edu server?** It is an interactive & time-sharing system. **Which one of the following is not shared by threads that share the same address space?** Their stacks & program counters. **Which system call returns the process ID of a terminated child?** Wait(). **The time required to create a new thread in an existing process is:** Less than the time required to create a new process. **The execv() system call specifies which new program a process should execute.** True. **The execv() system call creates a new process.** False. **Which hardware mechanism allows a device to notify the CPU of an event?** A. Interrupts. **Which of these events can move a process from the running state to the ready state?** A timer interrupt. The arrival in the ready state of a higher priority process. **Which of these events can move a process from the running state to the blocked state?** The process performs a system call. **In which queue is a newly created process initially put?** Ready queue. **Memory protection is normally done through privileged instructions.** False. **Delaying disk—or SSD—writes.** May result in lost data if the system crashes. **Which of the following statements does not apply to microkernels?** They are faster than most other kernel organizations. **A process in the ready state can only move from that state to the:** Running State. **Which of the following actions are the normal result of a system call?** An interrupt occurs. **What is the default action a Linux process takes when it receives a signal from another process?** It terminates.

The goal of the suspend state in a five-state process model with one suspend state is to making room in main memory for new process by moving the process in the ready state to virtual memory. False. **The main objective of a time-sharing system is to reduce the response time?** True. **In a pure ULT strategy, a multithreaded application cannot take advantage of multiprocessing?** True. **Does a small cache block size improve the hit ratio of the principal of locality.** False. **Does a system call generate a change of mode (from KERNAL MODE TO USER MODE).** False. **In an interrupt-driven i/o call, the processor is the resource that handles the transfer of information between the I/o devices and memory?** True. **Select the thread implementation where threads must use the scheduler provided by the OS.** Kernel-level threads. **Select the process state where process reside in main memory and can be chosen by the scheduler to be executed.** Running. **The kernel structure that includes virtually all the OS functionalities is a.** Monolithic Kernel. **What type of interrupt is a division by zero.** Program. **Select the multiple interrupt handling mechanism that disables interrupts while an interrupt is being processed.** Multi-Interrupts; Sequential. **The process that executes the instruction after if((pid=fork())==0).** Child Process.