Quizlet Chapter 3

Study online at quizlet.com/_u4euq

1.	**Address Space**	Child duplicate of parent Child has a program loaded into it
2.	**advantages of process cooperation**	Information sharing Computation speed-up Modularity Convenience
3.	**All ports below 1024**	are well known, used for standard services
4.	**As a process executes, it changes**	state
5.	**Blocking**	is considered synchronous Blocking send -- the sender is blocked until the message is received Blocking receive -- the receiver is blocked until a message is available
6.	**bounded-buffer**	assumes that there is a fixed buffer size
7.	**Buffering**	Queue of messages attached to the link. implemented in one of three ways 1. Zero capacity - no messages are queued on a link. Sender must wait for receiver (rendezvous) 2. Bounded capacity - finite length of n messages Sender must wait if link full 3. Unbounded capacity - infinite length Sender never waits
8.	**cascading termination**	All children, grandchildren, etc. are terminated.
9.	**Communication**	onsists between a pair of sockets
10.	**Communications in Client-Server Systems**	Sockets Remote Procedure Calls Pipes Remote Method Invocation (Java)
11.	**Concatenation of IP address and port**	a number included at start of message packet to differentiate network services on a host
12.	**Context Switch**	When CPU switches to another process, the system must save the state of the old process and load the saved state for the new process via a context switch Context of a process represented in the PCB Context-switch time is overhead; the system does no useful work while switching The more complex the OS and the PCB ->the longer the context switch Time dependent on hardware support Some hardware provides multiple sets of registers per CPU multiple contexts loaded at once
13.	**Cooperating process**	can affect or be affected by the execution of another process
14.	**CPU-bound process**	spends more time doing computations; few very long CPU bursts
15.	**Direct Communication**	Processes must name each other explicitly: send(P, message) - send a message to process P receive(Q, message) - receive a message from process Q
16.	**Execution options**	Parent and children execute concurrently Parent waits until children terminate
17.	**Google Chrome Browser is multiprocess with 3 different types of processes:**	Browser process manages user interface, disk and network I/O Renderer process renders web pages, deals with HTML, Javascript. A new renderer created for each website opened Runs in sandbox restricting disk and network I/O, minimizing effect of security exploits Plug-in process for each type of plug-in
18.	**If processes P and Q wish to communicate,**	hey need to: Establish a communication link between them Exchange messages via send/receive

19.	**Implementation of communication link**	Physical: Shared memory Hardware bus Network Logical: Direct or indirect Synchronous or asynchronous Automatic or explicit buffering
20.	**Independent process**	cannot affect or be affected by the execution of another process
21.	**Indirect Communication operations**	create a new mailbox (port) send and receive messages through mailbox destroy a mailbox
22.	**Indirect Communication operations 2**	Primitives are defined as: send(A, message) - send a message to mailbox A receive(A, message) - receive a message from mailbox A
23.	**Interprocess Communication**	Processes within a system may be independent or cooperating Cooperating process can affect or be affected by other processes, including sharing data Reasons for cooperating processes: Information sharing Computation speedup Modularity Convenience Cooperating processes need interprocess communication (IPC)
24.	**I/O-bound process**	spends more time doing I/O than computations, many short CPU bursts
25.	**IPC Systems - Windows**	Message-passing centric via advanced local procedure call (LPC) facility Only works between processes on the same system Uses ports (like mailboxes) to establish and maintain communication channels Communication works as follows: The client opens a handle to the subsystem's connection port object The client sends a connection request The server creates two private communication ports and returns the handle to one of them to the client The client and server use the corresponding port handle to send messages or callbacks and to listen for replies.
26.	**Long-term scheduler (or job scheduler)**	selects which processes should be brought into the ready queue Long-term scheduler is invoked infrequently (seconds, minutes) (may be slow) The long-term scheduler controls the degree of multiprogramming
27.	**Long-term scheduler strives for a**	good process mix
28.	**Mach communication is message based**	Even system calls are messages Each task gets two mailboxes at creation- Kernel and Notify Only three system calls needed for message transfer msg_send(), msg_receive(), msg_rpc() Mailboxes needed for commuication, created via port_allocate() Send and receive are flexible, for example four options if mailbox full: Wait indefinitely Wait at most n milliseconds Return immediately Temporarily cache a message
29.	**Mailbox sharing**	P1, P2, and P3 share mailbox A P1, sends; P2 and P3 receive Who gets the message?
30.	**Medium-term scheduler**	can be added if degree of multiple programming needs to decrease Remove process from memory, store on disk, bring back in from disk to continue execution: swapping
31.	**Message Passing**	Mechanism for processes to communicate and to synchronize their actions Message system - processes communicate with each other without resorting to shared variables IPC facility provides two operations: send(message) receive(message) The message size is either fixed or variable
32.	**Messages are directed and received from mailboxes (also referred to as ports)**	Each mailbox has a unique id Processes can communicate only if they share a mailbox

33.	**Multiple parts of process**	The program code, also called text section Current activity including program counter, processor registers Stack containing temporary data Function parameters, return addresses, local variables Data section containing global variables Heap containing memory dynamically allocated during run time
34.	**Multitasking in Mobile Systems**	Some mobile systems (e.g., early version of iOS) allow only one process to run, others suspended Due to screen real estate, user interface limits, iOS provides for a Single foreground process- controlled via user interface Multiple background processes- in memory, running, but not on the display, and with limits Limits include single, short task, receiving notification of events, specific long-running tasks like audio playback Android runs foreground and background, with fewer limits Background process uses a service to perform tasks Service can keep running even if background process is suspended Service has no user interface, small memory use
35.	**Named Pipes**	Named Pipes are more powerful than ordinary pipes Communication is bidirectional No parent-child relationship is necessary between the communicating processes Several processes can use the named pipe for communication Provided on both UNIX and Windows system
36.	**new**	The process is being created
37.	**Non-blocking**	is considered asynchronous Non-blocking send -- the sender sends the message and continues Non-blocking receive -- the receiver receives: A valid message, or Null message
38.	**One program can be several processes**	Consider multiple users executing the same program Single program can also spawn many processes
39.	**An operating system executes a variety of program**	Batch system - jobs Time-shared systems - user programs or tasks
40.	**Ordinary Pipes**	Ordinary Pipes allow communication in standard producer-consumer style Producer writes to one end (the write-end of the pipe) Consumer reads from the other end (the read-end of the pipe) Ordinary pipes are therefore unidirectional Require parent-child relationship between communicating processes Windows calls these anonymous pipes
41.	**orphan**	If parent terminated without invoking wait
42.	**Paradigm for cooperating processes**	producer process produces information that is consumed by a consumer process
43.	**Parent may terminate the execution of children processes using the abort()system call. Some reasons for doing so:**	Child has exceeded allocated resources Task assigned to child is no longer required The parent is exiting and the operating systems does not allow a child to continue if its parent terminates
44.	**Pipes**	Acts as a conduit allowing two processes to communicate Issues: Is communication unidirectional or bidirectional? In the case of two-way communication, is it half or full-duplex? Must there exist a relationship (i.e., parent-child) between the communicating processes? Can the pipes be used over a network? Ordinary pipes - cannot be accessed from outside the process that created it. Typically, a parent process creates a pipe and uses it to communicate with a child process that it created. Named pipes - can be accessed without a parent-child relationship.

45. **POSIX Shared Memory**	Process first creates shared memory segment shm_fd = shm_open(name, O_CREAT | O_RDWR, 0666);
	Also used to open an existing segment to share it Set the size of the object ftruncate(shm_fd, SIZE); //SIZE=4096
	Memory map the shared memory object ptr = mmap(0, SIZE, PROT_WRITE, MAP_SHARED, shm_fd, 0);
	Now the process could write to the shared memory
	sprintf(ptr, "%s", "Writing to shared memory");

46. **Process**	a program in execution; process execution must progress in sequential fashion
47. **Process Control Block (PCB)**	Information associated with each process (also called task control block) Process state - running, waiting, etc Program counter - location of instruction to next execute CPU registers - contents of all process-centric registers CPU scheduling information- priorities, scheduling queue pointers Memory-management information - memory allocated to the process Accounting information - CPU used, clock time elapsed since start, time limits I/O status information - I/O devices allocated to process, list of open files
48. **Process Creation**	Parent process create children processes, which, in turn create other processes, forming a tree of processes Generally, process identified and managed via a process identifier (pid)
49. **Process Scheduling**	Maximize CPU use, quickly switch processes 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 Ready queue - set of all processes residing in main memory, ready and waiting to execute Device queues - set of processes waiting for an I/O device Processes migrate among the various queues
50. **Process Termination**	Process executes last statement and then asks the operating system to delete it using the exit()system call. Returns status data from child to parent (via wait()) Process'resources are deallocated by operating system
51. **Process Termination 2**	Some operating systems do not allow child to exist if its parent has terminated. If a process terminates, then all its children must also be terminated.
52. **Program becomes process**	when executable file loaded into memory
53. **Program is passive entity stored on disk (executable file**	but process is active
54. **Properties of communication link**	Links are established automatically A link is associated with exactly one pair of communicating processes Between each pair there exists exactly one link The link may be unidirectional, but is usually bi-directional
55. **Properties of communication link 2**	Link established only if processes share a common mailbox A link may be associated with many processes Each pair of processes may share several communication links Link may be unidirectional or bi-directional
56. **Queueing diagram**	represents queues, resources, flows
57. **ready**	the process is waiting to be assigned to a processor
58. **Remote procedure call (RPC)**	abstracts procedure calls between processes on networked systems Again uses ports for service differentiation
59. **Remote Procedure calls 2**	Data representation handled via External Data Representation (XDL) format to account for different architectures Big-endian and little-endian Remote communication has more failure scenarios than local Messages can be delivered exactly once rather than at most once OS typically provides a rendezvous (or matchmaker) service to connect client and server

60.	**rendezvous**	If both send and receive are blocking,
61.	**Resource sharing options**	Parent and children share all resources Children share subset of parent's resources Parent and child share no resources
62.	**running**	Instructions are being executed
63.	**Shared memory**	An area of memory shared among the processes that wish to communicate The communication is under the control of the users processes not the operating system. Major issues is to provide mechanism that will allow the user processes to synchronize their actions when they access shared memory.
64.	**Short-term scheduler (or CPU scheduler)**	elects which process should be executed next and allocates CPU Sometimes the only scheduler in a system Short-term scheduler is invoked frequently (milliseconds) (must be fast)
65.	**socket**	is defined as an endpoint for communication
66.	**The socket 161.25.19.8:1625**	refers to port 1625 on host 161.25.19.8
67.	**Sockets in Java**	Three types of sockets Connection-oriented (TCP) Connectionless (UDP) MulticastSocket class- data can be sent to multiple recipients Consider this "Date" server:
68.	**Solutions**	Allow a link to be associated with at most two processes Allow only one process at a time to execute a receive operation Allow the system to select arbitrarily the receiver. Sender is notified who the receiver was.
69.	**Special IP address 127.0.0.1**	loopback) to refer to system on which process is running
70.	**Stubs**	client-side proxy for the actual procedure on the server
71.	**Stubs 2**	The client-side stub locates the server and marshalls the parameters The server-side stub receives this message, unpacks the marshalled parameters, and performs the procedure on the server On Windows, stub code compile from specification written in Microsoft Interface Definition Language (MIDL)
72.	**terminiated**	the process has finished execution
73.	**Two models of IPC**	Shared memory Message passing
74.	**unbounded-buffer**	places no practical limit on the size of the buffer
75.	**UNIX examples of address space**	fork()system call creates new process exec()system call used after a fork()to replace the process'memory space with a new program
76.	**waiting**	the process is waiting for some event to occur
77.	**wait()system call**	The parent process may wait for termination of a child process by using the wait()system call. The call returns status information and the pid of the terminated process pid = wait(&status);
78.	**Web broswers**	Many web browsers ran as single process (some still do) If one web site causes trouble, entire browser can hang or crash
79.	**zombie**	If no parent waiting (did not invoke wait())