**E.G.S.PILLAY ENGINEERING COLLEGE, NAGAPATTINAM**

**DEPARTMENT OF MCA**

**MC7404-NETWORK PROGRAMMING**

**CYCLE TEST-1 KEY DATE: 26.02.2016**

**UNIT 1-PART-A**

1. Describe layered model of TCP/IP protocol.( May/June 2013)



1. Difference between move and copy a file(August 2013)

Cp old\_filename destination\_filename- a copy of the file will be moved to the destination, source remains in same place.

Mv old\_file new\_file- there won’t be any copy at the source , it is completely moved to destination file.

1. Compare signals and interrupts in UNIX.( May/June 2014)

Signals are software interrupts. Most nontrivial application programs need to deal with signals.

Signals provide a way of handling asynchronous events: a user at a terminal typing the interrupt key

to stop a program or the next program in a pipeline terminating prematurely.

1. What are the common shells in use in UNIX system? How does the system decide which shell is to be executed? May/June 2015)

A *shell* is a command-line interpreter that reads user input and executes commands. The user input to a shell is normally from the terminal (an interactive shell) or sometimes from a file (called a *shell script*).

* Bourne shell
* Bourne-again shell
* C shell
* Korn shell
* TENEX C shell

The system knows which shell to execute for us from the final field in our entry in the password file.

1. What is socket? ( May/June 2013,2014)

A *socket* is one endpoint of a two-way communication link between two programs running on the network. A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent to.

An endpoint is a combination of an IP address and a port number. Every TCP connection can be uniquely identified by its two endpoints. That way you can have multiple connections between your host and the server.

1. What is the purpose of concurrent server? .( May/June 2013)

*concurrent server*, one that handles multiple clients at the same time. The simplest

technique for a concurrent server is to call the Unix fork function (Section 4.7), creating

one child process for each client.

1. The INADDR-ANY & INADDR-BROADCAST constants defined by <netinet/in.h> header are in host byte order. Justify the statement. ( May/June 2015)

With IPv4, the *wildcard* address is specified by the constant INADDR\_ANY, whose value is normally 0. This tells the kernel to choose the IP address.

The value of INADDR\_ANY (0) is the same in either network or host byte order, so the use of htonl is not really required. But, since all the INADDR\_constants defined by the <netinet/in.h> header are defined in host byte order, we should use htonl with any of these constants.

1. List the eight ways of process termination. ( Feb/March 2014)

1. Return from main

**2.** Calling exit

**3.** Calling \_exit or \_Exit

**4.** Return of the last thread from its start routine

**5.** Calling pthread\_exit from the last thread

Abnormal termination occurs in three ways:

**6.** Calling abort

**7.** Receipt of a signal

8. Response of the last thread to a cancellation request.

1. Define reliable signal. ( Feb/March 2014)

signal is *generated* for a process (or sent to a process) when the event that causes the signal occurs. The event could be a hardware exception (e.g., divide by 0), a software condition (e.g., an alarm timer expiring), a terminal-generated signal, or a call to the kill function. When the signal is generated, the kernel usually sets a flag of some form in the process table.

**PART-B**

**11 a. Write a detailed note on TCP Protocol.**

.

11. b.**How do you establish Interprocess communication in UNIX? Give example**

**Introduction**

Pipes: A pipe is created by calling the pipe function.

**#include <unistd.h>**

**int pipe(int *filedes[2]*);**

**Returns: 0 if OK, 1 on error**

**Coprocesses:**

A UNIX system filter is a program that reads from standard input and writes to standard output. Filters are normally connected linearly in shell pipelines. A filter becomes a *coprocess* when the same program generates the filter's input and reads the filter's output. The Korn shell provides coprocesses [Bolsky and Korn 1995].

**FIFOs:**

FIFOs are sometimes called named pipes. Pipes can be used only between related processes when a common ancestor has created the pipe. Creating a FIFO is similar to creating a file. Indeed, the *pathname* for a FIFO exists in the file system.

**Semaphores:**

A semaphore is a counter used to provide access to a shared data object for multiple processes. The Single UNIX Specification includes an alternate set of semaphore interfaces in the semaphore option of its real-time extensions. We do not discuss these interfaces in this text. To obtain a shared resource, a process needs to do the following:

**Shared Memory:**

**struct shmid\_ds {**

**struct ipc\_perm shm\_perm; /\* see Section 15.6.2 \*/**

**size\_t shm\_segsz; /\* size of segment in bytes \*/**

**pid\_t shm\_lpid; /\* pid of last shmop() \*/**

**pid\_t shm\_cpid; /\* pid of creator \*/**

**shmatt\_t shm\_nattch; /\* number of current attaches \*/**

**time\_t shm\_atime; /\* last-attach time \*/**

**time\_t shm\_dtime; /\* last-detach time \*/**

**time\_t shm\_ctime; /\* last-change time \*/ };**

**The first function called is usually shmget, to obtain a shared memory identifier.**

**#include <sys/shm.h>**

**int shmget(key\_t *key*, size\_t *size*, int *flag*);**

**Returns: shared memory ID if OK, 1 on error**

12.b.What are the different file types available in UNIX? Explain with example.(Feb/march 2014)

1. **Regular file**.
2. **Directory file**
3. **Block special file**.
4. **Character special file**.
5. **FIFO**.
6. **Socket**.
7. **Symbolic link**.
8. **File type**

12.a. Write about files and directories in Unix

stat, fstat, and lstat Functions

**#include <sys/stat.h>**

**int stat(const char \*restrict *pathname*, struct**

**stat \*restrict *buf*);**

**int fstat(int *filedes*, struct stat \**buf*);**

**int lstat(const char \*restrict *pathname*, struct**

**stat \*restrict *buf*);**

**All three return: 0 if OK, 1 on error**

**Set-User-ID and Set-Group-ID**

* real user ID,real group ID :who we really are
* effective user ID , effective group ID, supplementary group IDs: used for file access permission , checks

**File Access Permissions**

**st\_mode mask Meaning**

**S\_IRUSR user-read**

**S\_IWUSR user-write**

**S\_IXUSR userexecute**

**Ownership of New Files and Directories:** implementation to choose one of the following options to determine the group ID of a new file.

1. The group ID of a new file can be the effective group ID of the process.
2. The group ID of a new file can be the group ID of the directory in which the file is being created. The Linux ext2 and ext3 file systems allow the choice between these two POSIX.1 options

**access Function:** The access function bases its tests on the real user and group IDs.

**#include <unistd.h>**

**int access(const char \**pathname*, int *mode*);**

**Returns: 0 if OK, 1 on error**

**Symbolic Links**

13.b.Explain architecture of unix in details.(Aug 2011)

 

It is responsible for scheduling running of user and other processes. It is responsible for allocating memory. It is responsible for managing the swapping between memory and disk. It is responsible for moving data to and from the peripherals. It receives service requests from the processes and honours them.

14.a. Explain the socket address structure.

**Socket Address Structures**

Most socket functions require a pointer to a socket address structure as an argument. Each

supported protocol suite defines its own socket address structure. The names of these structures

begin with sockaddr\_ and end with a unique suffix for each protocol suite.

**IPv4 Socket Address Structure**

An IPv4 socket address structure, commonly called an "Internet socket address structure," is

named sockaddr\_in and is defined by including the <netinet/in.h> header. Figure 3.1 shows the

POSIX definition.

**Generic Socket Address Structure**

**struct sockaddr {**

**uint8\_t sa\_len;**

**sa\_family\_t sa\_family; /\* address family: AF\_xxx value \*/**

**char sa\_data[14]; /\* protocol-specific address \*/**

**};**

**New Generic Socket Address Structure**

1. B.Explain the following function in socket programming. (i) connect () (ii) bind() (iii) listen() (iv) socket() (v) accept().( May/June 2014)
2. **socket Function**

**#include <sys/socket.h>**

**int socket (int *family*, int *type*, int *protocol*);**

**Returns: non-negative descriptor if OK, -1 on error**

1. **connect Function**

**#include <sys/socket.h>**

**int connect(int *sockfd*, const struct sockaddr \**servaddr*, socklen\_t *addrlen*);**

**Returns: 0 if OK, -1 on error**

 **(iii) bind Function**

**#include <sys/socket.h>**

**int bind (int *sockfd*, const struct sockaddr \**myaddr*, socklen\_t *addrlen*);**

**Returns: 0 if OK,-1 on error**

1. **listen Function**

**#include <sys/socket.h>**

**#int listen (int *sockfd*, int *backlog*);**

**Returns: 0 if OK, -1 on error**

1. **accept Function**

**#include <sys/socket.h>**

**int accept (int *sockfd*, struct sockaddr \**cliaddr*, socklen\_t \**addrlen*);**

**Returns: non-negative descriptor if OK, -1 on error**

15.a.Explain the steps involved in terminal login.(Feb/March 2014)

* + **BSD Terminal Logins**

 



1. B. Write short on address conversion function.

 **inet\_aton, inet\_addr, and inet\_ntoa Functions**

**#include <arpa/inet.h>**

**int inet\_aton(const char \**strptr*, struct in\_addr \**addrptr*);**

**Returns: 1 if string was valid, 0 on error**

**in\_addr\_t inet\_addr(const char \**strptr*);**

**Returns: 32-bit binary network byte ordered IPv4 address; INADDR\_NONE if error**

**char \*inet\_ntoa(struct in\_addr *inaddr*);**

**Returns: pointer to dotted-decimal string**

