UNIVERSITY OF TORONTO
Faculty of Arts and Science

APRIL EXAMINATIONS 2001

CSC 209H1S

Duration - 3 hours

Examination Aids: none

- **Make sure that your test has 12 pages (including this one).**
- Write your answers in the spaces provided.
- You will be rewarded for concise, well thought-out answers, rather than long ones. Write legibly.
- Read through the exam first before starting.
- The last page contains a list of C prototypes you may find useful.

Last Name: ____________________ **Given Names:** ____________________

Student #: ____________________

1.	_____	/	7
2.	_____	/	6
3.	_____	/	13
4.	_____	/	6
5.	_____	/	8
6.	_____	/	13
7.	_____	/	16
8.	_____	/	7
9.	_____	/	12
10.	_____	/	14
TOTAL:	_____	/	102

Question 1 [7 marks in total]

Circle the correct answer for the following questions:

a. TRUE FALSE The Unix command “cd ~bob” changes the current working directory to the home directory of the user named bob.

b. TRUE FALSE Given that Perl has an interface for Unix sockets, a program written in Perl can communicate with a program written in C across sockets.

c. TRUE FALSE After a call to fork() the parent always executes before the child.

d. TRUE FALSE Version control systems such as CVS make it possible to undo changes to a file.

e. TRUE FALSE A call to `semop()` blocks until the value of the semaphore variable is greater than 0.

f. TRUE FALSE You should always program using a low-level language because then you have control over every detail.

g. TRUE FALSE A process using `select()` for multiplexing I/O will never block.

Question 2 [6 marks in total]

a. [3 marks] Circle the commands that cause a new process with a new process ID to be created when run in a Bourne shell script. (` is a backquote)

```
echo ls *.c                   cat file

d = `date`                    echo `ls *.c`

read invar                    set d
```

b. [3 marks] Circle the C function calls that cause a new process with a new process ID to be created.

```
execv("pname", args);         fp = popen("uptime", "r")

fp = fopen("infile","r")      result = pipe(fd);

pid = fork()                  num = fread(buf, n, 1 ,fp)
```

Question 3 [13 marks in total]

a. [2 marks] What is a critical section?

b. [2 marks] Define deadlock, and give an example of where deadlock might arise.

c. [2 marks] Define starvation, and give an example of how it might occur.

d. [2 marks] Name two different socket types.

e. [2 marks] What is wrong with the following code? Assume all variables are properly declared, and `pid` is the process ID of a child of the process running this code.

```
result = waitpid(pid, &status, WNOHANG);
if( WIFEXITED(status) )
    printf("Child %d has exited\n", pid);
```

f. [3 marks] Explain what happens when the following code is executed. In particular, describe what happens to the shared memory segment. If an error occurs, explain the error.

```
char *sptr;
int shmID = shmget(IPC_PRIVATE, size, (SHM_R|SHM_W)) < 0);
sptr = (char *)shmat(shmId,0,0);
shmctl(shmId, IPC_RMID, NULL);
sptr[0] = 'a';
```

Question 4 [6 marks in total]

Write a Bourne shell script that will move all files ending in .mp3 from the directory “download” to the directory “music”. Assume that “download” and “music” exist and are subdirectories of the current directory. Your script must not produce errors if the file names have spaces in them. Also, your script must not use backquoted commands.

Question 5 [8 marks in total]

In lecture we talked about 5 different I/O models. One of them was asynchronous I/O. Name the other 4, and give one advantage of each model.

Question 6 [13 marks in total]

Write a short C program that creates two children. The first child executes a function `void DoSend(int writefd)` and then exits immediately. The second child executes a function `void DoFilter(int readfd, int writefd)` and then exits. The parent executes a function `void DoReceive(int readfd)`. The program creates two pipes. One pipe connects from the first child to the second child, and the other pipe connects from the second child to the parent.

a. [2 marks] Draw a diagram that illustrates the program described above. The diagram should contain a labeled box for each process, connections for each of the pipes, and labels for the file descriptors.

b. [1 marks] Does the parent need to call `wait()` before calling `DoReceive(int readfd)`?

c. [10 marks] Write the C program described above. Remember to close unused file descriptors. Do not write the bodies of `DoSend()`, `DoFilter()`, or `DoReceive()`.

(Extra page if you need it.)

Question 7 [16 marks in total]

a. [8 marks] List and explain the purpose of the 4 system calls that are used to set up a TCP socket for a server.

b. [8 marks] Write a client program in C that will send to a server, a countdown message once per second for 10 seconds, and a final "blastoff" message before exiting. The client communicates with the server across a TCP socket connection, does not receive any messages from the server. The server's IP address is a command line parameter. The port number of the server is defined in the global variable `PORT`. You will find the following two lines of code helpful:

```
struct sockaddr_in servaddr;
inet_pton(AF_INET, argv[1], &servaddr.sin_addr);
```

The client will send the server the following messages, one every second:

`"10" "9" "8" "7" "6" "5" "4" "3" "2" "1" "blastoff"`

(Continue your answer to Question 7 on this page.)

Question 8 [7 marks in total]

Consider the following program. The function `int power(int a, int b)` returns a^b.

```
int sum, childresult;

int main() {
    int pid, i, status;
    sum = 0;
    childresult = 0;

    for(i = 1; i < 3; i++) {
        pid = fork();
        if(pid == 0) {
            childresult = power(i, 3);
            printf("Child: %d\n", childresult);
            exit();
        } else {
            wait(&status);
            printf("Parent: %d\n", childresult);
            sum += childresult;
        }
    }
    printf("total = %d\n", sum);
}
```

a. [1 marks] How many processes are created including the initial process? ________

b. [1 marks] How many times is `wait()` called? ________

c. [5 marks] Write the output that is produced by this program in a valid order.

Question 9 [12 marks in total]

Write a C program that computes $sum = 1^3 + 2^3 + 3^3 + ... + N^3$. Your program will create N threads such that thread i computes i^3 and adds its result to the global sum. The initial thread will wait for all of the other threads to complete, and will print the sum.

Each thread will be created to call a function `do_cube()` and will take i as an argument. Remember that all threads execute in the same address space, so you need to be careful how arguments are passed, and how global variables are updated.

Question 10 [14 marks in total]

a. [2 marks] Write a Perl regular expression that matches lines from the first column but not from the second. The regular expression should be as general as possible.

`apple 5 @ 0.50  2.50`
`peas            1.25`
`soup  100 @ .99  99.00`
	`potatoes          10.75 15.40`
	`candle      5    20.49`

b. [6 marks] Given the following Perl regular expressions, circle the strings to the right of each expression that match that expression.

`/[sS]um\s*\w+/`	`"summary"`	`"Sum total"`	`"Sum =1"`	`"sSum"`
`/^[ag](209|0|7)\w+/`	`"a209nico"`	`"ta209and"`	`"a0reid"`	`"g9tehpoh"`
`/([\d-]+)\s+(\d+-\d+)/`	`"9-9 9-9"`	`"-11 11"`	`"8-1-2 8-12"`	`"8-12 8-1-2"`

c. [6 marks] Write a Perl script that reads transactions from a file and applies them to an array of numbers. After all transactions have been applied, print the index and the value of the elements of the array that are less than 0.

There is one transaction on each line of the file, and a transaction is composed of 2 numbers. The first number is the index into an array, and the second number is a value to be added to the current value at the index into the array. For example, the transaction "`6 60`" means "add 60 to the array element at index 6" and "`3 -10`" means "add -10 to array element at index 3".

```
open(TRANS, "<transactions") or die "Couldn't open transactions\n";
```

Function Prototypes:

```
int accept(int soc, struct sockaddr *addr, int addrlen)
int bind(int soc, struct sockaddr *addr, int addrlen)
int close(int fd)
int connect(int soc, struct sockaddr addr, int addrlen)
int dup2(int fd, int oldfd)
int execl(const char *path, char *argv0, ..., (char *)0)
int execle(const char *path, char *argv0, ..., (char *)0, const char *envp[])
int execlp(const char *file, char *argv0, ..., (char *)0)
int execv(const char *path, char *argv[])
int execve(const char *path, char *argv[], const char *envp[])
int execvp(const char *file, char *argv[])
int fclose(FILE *stream)
int FD_ISSET(int fd, fd_set &fds)
void FD_SET(int fd, fd_set &fds)
void FD_CLR(int fd, fd_set &fds)
void FD_ZERO(fd_set &fds)
char *fgets(char *s, int n, FILE *stream)
int fileno(FILE *stream)
pid_t fork(void)
FILE *fopen(const char *file, const char *mode)
int fprintf(FILE *stream, const char *format, ...)
int kill(int pid, int signo)
int listen(int soc, int n)
int open(const char *path, int oflag)
int pipe(int filedes[2])
int pclose(FILE *stream)
int pthread_mutex_destroy(pthread_mutex_t *mutex)
int pthread_mutex_lock(pthread_mutex_t *mutex)
int pthread_mutex_unlock(pthread_mutex *mutex)
FILE *popen(char *cmdstr, char *mode)
int pthread_mutex_init(pthread_mutex_t *mutex,
                     const pthread_mutex_attr_t *attr)
ssize_t Readline(int filedes, void *buf, size_t maxlen);
ssize_t Readn(int filedes, void *buf, size_t nbytes);
int select(int maxfdp1, fd_set *readfds, fd_set *writefds,
           fd_set *exceptfds, struct timeval *timeout)
int semctl(key_t key, int semnum, int cmd, union semun arg))
int semget(key_t key, int nsems, int semflags)
int semop(int semId, struct semops *sem_ops, int nops)
void *shmat ( int shmid, const void *shmaddr, int shmflg )
int shmget(key_t key, int size, int shmflg)
int socket(int family, int type, int protocol)
int sprintf(char *s, const char *format, ...)
int wait(int &status)
int waitpid(int pid, int *stat, int options)
void Writen(int filedes, const void *buf, size_t nbytes);
```

```
struct sockaddr_in {
  sa_family_t          sin_family;
  unsigned short int   sin_port;
  struct in_addr       sin_addr;
  unsigned char        pad[8];        /* Unused */
};
```