PLEASE HAND IN

PLEASE HAND IN

UNIVERSITY OF TORONTO
Faculty of Arts and Science

APRIL-MAY EXAMINATIONS 2001

CSC 209H1 S
Duration — 3 hours

Examination Aids: ***None***

Student Number: _ _ _ _ _ _ _ _ _ _

Last Name: _____

First Name: _____

Lecture Section: _____

Lecturer: Reid

---

*Do **not** turn this page until you have received the signal to start.*
(In the meantime, please fill out the identification section above, and read the instructions below *carefully*.)

This final examination consists of 9 questions on 16 pages (including this one), *When you receive the signal to start, please make sure that your copy of the examination is complete.* Answer each question directly on the examination paper, in the space provided.

Be aware that concise, well thought-out answers will be rewarded over long rambling ones. Also, unreadable answers will be given zero so write legibly.

You do **not** need to include header files or do error checking in C programs except where specifically mentioned in a question. The last two pages of this exam contain a list of C function prototypes structs, and some Perl, Python and Bourne shell details.

1: _____/10

2: _____/ 6

3: _____/12

4: _____/ 8

5: _____/ 8

6: _____/14

7: _____/10

8: _____/12

9: _____/10

TOTAL: _____/90

Good Luck!

Question 1. [10 MARKS]

Part (a) [2 MARKS]

In assignment 3, child processes wrote to the parent through pipes. Explain how the parent knew when it could start reading from the pipes.

Part (b) [2 MARKS]

Explain why is it important to close the ends of pipes when they are no longer needed. What problem can occur if the write end of a pipe is not closed?

Part (c) [2 MARKS]

Give one reason why a programmer would want to block a signal.

Part (d) [2 MARKS]

Give one reason why a programmer would want to alter the behaviour of a signal by installing a signal handling function.

Part (e) [2 MARKS]

Explain what the zombie state of a process is used for.

Question 2. [6 MARKS]

Part (a) [3 MARKS]

Scripting languages such as Perl and Python are frequently used in web programming. Explain why scripting languages might be preferred over C for this type of programming.

Part (b) [3 MARKS]

Given the following C program, circle **all** valid output sequences. Note that 0 or more of the sequences may be valid.

```
int main() {
    int num, status;
    num = 10;
    printf("A: %d\n", num);

    if( fork() == 0 ) {
        num = 24;
        printf("B: %d\n", num);
    } else {
        wait(&status);
    }
    printf("C: %d\n", num);
}
```

A: 10 B: 24 C: 24 C: 24	A: 10 B: 24 C: 24 C: 10	A: 10 B: 24 C: 10 C: 24	A: 10 B: 24 C: 10	A: 10 B: 24 C: 24
A: 10 C: 24 B: 24 C: 24	A: 10 C: 10 B: 24 C: 24	A: 10 C: 10 B: 24	A: 10 C: 24 B: 24	

Question 3. [12 MARKS]

Part (a) [1 MARK]

Write a shell command that will concatenate all of the files in the current working directory that end in .h and put them into a file called `headers`.

Part (b) [2 MARKS]

Consider the following two shell programs:

testsbs

```
#!/bin/sh
if [ -f .sbs ]
then
        echo found
        exit 0
fi
exit 1
```

runtests

```
#!/bin/sh
# ` is a backquote

result=`testsbs`
```

When `runtests` is executed, the value of `$result` if there *is* a file called .sbs in the current working directory is

_____.

When `runtests` is executed, the value of `$result` if there *is not* a file called .sbs in the current working directory is

_____.

Part (c) [5 MARKS]

If the contents of the current working directory are

```
a.c a.o bb qq
```

and the following Bourne shell commands are executed, what is printed? (` is a backquote and ' is a single quote.)

```
cmd="ls *"
set `$cmd`

echo $cmd      ________________________

echo $1        ________________________

echo ??        ________________________

echo '??'      ________________________

echo "$cmd"    ________________________
```

Part (d) [4 MARKS]

Assuming that a Perl or Python program is stored in a file called `getvotes`, describe the two different ways the program could be run. Give any conditions necessary to allow the program to be run, and explain what the user must type.

Question 4. [8 MARKS]

Given the following Makefile and the contents of the current working directory:

```
CFLAGS = -g -Wall
OBJS = main.o a.o b.o

default: main

main : $(OBJS)
        gcc $(CFLAGS) -o main $(OBJS)

a.o: a.c
        gcc -c a.c
b.o: b.c
        gcc -c b.c
main.o: main.c
        gcc -c main.c
```

```
-rw-r--r--  reid 177 Mar 25 11:24 Makefile
-rw-r--r--  reid 118 Mar 25 11:06 a.c
-rw-r--r--  reid  17 Mar 25 11:06 a.h
-rw-r--r--  reid  72 Mar 25 11:06 b.c
-rw-r--r--  reid  32 Mar 25 11:06 b.h
-rw-r--r--  reid 137 Mar 25 11:08 main.c
```

Part (a) [2 MARKS]

Write the commands that would be executed when the user types `make`.

Part (b) [2 MARKS]

Suppose the user then modifies `main.c`, and then types `make`. Write the commands that are executed.

Part (c) [2 MARKS]

Write an additional rule with the target name `clean` that removes all `.o` files and the executable `main`.

Part (d) [2 MARKS]

If the user modifies `a.h` and then calls `make`, nothing will be recompiled. Change or add a rule so that `a.o` will be re-generated (recompiled) when `a.h` is modified.

Question 5. [8 MARKS]

Write a Perl or Python program that takes as a command-line argument the name of a file containing lists of candidates and votes for each of the candidates. Your program will print the total number of votes cast, the number of votes for each candidate, and the percentage of the total vote each candidate received. (Don't worry about the significant digits.)

An input file and resulting output are given as an example:

Input file:

```
Jimmy Neutron: Boy Genius   15
Monsters, Inc.              100
Shrek                       150

Monsters, Inc.              200
Shrek                       102
Jimmy Neutron: Boy Genius   80

Shrek                       88
```

Output:

```
Total votes: 735
Shrek 340 46.25%
Monsters, Inc. 300 40.81%
Jimmy Neutron: Boy Genius 95 12.92%
```

Question 6. [14 MARKS]

Part (a) [2 MARKS]

In assignment 3, the child processes all accessed the same file. Explain why we did not need to use a semaphore to protect access to the file.

Part (b) [4 MARKS]

The Unix pipe system call sets up a pipe data structure in the kernel and synchronizes access to the pipe. There are three aspects of pipes that require some form of synchronization. One need for synchronization is to ensure that only one process at a time may access the pipe data structure. Describe the other two situations where synchronization is needed to ensure the correct operation of the pipe.

Part (c) [8 MARKS]

Complete the following pseudo-code functions to show how semaphores may be used to implement the synchronization necessary for the correct operation of pipes. You may assume the existence of the following functions:

`InitSem(int S, int value)`	Initialize the semaphore variable `S` to `value`
`acquire(int S)`	Perform the acquire or wait operation on the semaphore variable `S`.
`release(int S)`	Perform the release or signal operation on the semaphore variable `S`.
`read(buf)`	Read from some shared data structure into the buffer `buf`
`write(buf)`	Write to some shared data structure from the buffer `buf`

```
CreatePipe( int pipesize ){
    int mutex    /* semaphore variable */
    InitSem(mutex, 1)





}

ReadFromPipe(buf) {




    acquire(mutex)
    read(buf)
    release(mutex)




}

WriteToPipe(buf) {



    acquire(mutex)
    write(buf)
    release(mutex);



}
```

Question 7. [10 MARKS]

The following code is part of a program where a maximum of 3 clients connect to a server. The server prompts each client to enter a sequence of names and numbers. Clients execute a loop in which they read a prompt, write a name, read a prompt, and write a number. One problem with the program is the the server does not detect when a client closes a socket. However, there is a more serious problem with the implementation of this program. Assume that all of the set up and initialization is correct.

```
for ( ; ; ) {
    rset = allset;      /* structure assignment */
    nready = Select(maxfd+1, &rset, NULL, NULL, NULL);

    if (FD_ISSET(listenfd, &rset)) {    /* new client connection */
        clilen = sizeof(cliaddr);
        connfd = Accept(listenfd, (struct sockaddr *) &cliaddr, &clilen);
        printf("accepted a new client\n");

        Writen(connfd, "Enter a name", 13);
        for (i = 0; i < 3; i++)
            if (client[i] < 0) {
                client[i] = connfd; /* save descriptor */
                break;
            }
        if (i == 3) printf("too many clients");

        FD_SET(connfd, &allset);            /* add new descriptor to set */
        if (connfd > maxfd) maxfd = connfd; /* for select */
    }

    for (i = 0; i < 3; i++) {   /* check all clients for data */
        if (FD_ISSET(client[i], &rset)) {
            n = Readline(client[i], line, MAXLINE);
            printf ("Client name = %s\n", line);
            Writen(client[i], "Enter a number", 15);
            n = Readline(client[i], line, MAXLINE);
           printf ("Number = %d\n", atoi(line));
        }
    }
}
```

Part (a) [2 MARKS]

The major flaw of this program could cause it to hang (block) indefinitely. Explain what the problem is and under what circumstances it could cause the program to hang.

Part (b) [8 MARKS]

Fix the program so that it will not block unnecessarily. You do not need to rewrite all of the code, just indicate where the portions of the code you write belong in the provided code and cross out any code that should be deleted.

Question 8. [12 MARKS]

Part (a) [2 MARKS]

Name the two TCP socket functions that are used to initiate and complete the 3-way handshake.

Part (b) [2 MARKS]

What is the purpose of the 3-way handshake?

Part (c) [8 MARKS]

Complete the following socket **client** program that establishes a connection to a remote server given by `hostname` on the port `SERVER_PORT`. The server sends the client the address of another server to connect to and a message to write to that server and then closes the connection.

The client reads two messages from the first server. The first message is the address of the second server in network byte order. The length of the second message is between 1 and `BLOCK_SIZE` bytes. The client makes a connection to the second server using the address sent from the first server and the same port, and writes to the second server the bytes it received from the original server and closes the connection.

You must ensure that the client closes the socket, even when an error occurs. Recall that `htons()` must be called to convert a port into network byte order, and that the socket type will be `AF_INET`.

```
#define SERVER_PORT  30000
#define BLOCK_SIZE 1024

char *hostname = "penguin";

int main(void) {
    struct hostent *hp = gethostbyname(hostname);
    struct in_addr address;
```

(An extra page for the previous question.)

Question 9. [10 MARKS]

When creating programs to automatically test student assignments, we want to detect if a student's program gets into an infinite loop. Complete the C program below that takes the name of a second program to run as a command line argument. Your program will create a process to execute the second program (which itself takes no arguments). If the second program does not terminate after 10 seconds, the original process will terminate it. Your program will print out an error message if it must terminate the second process.

```
int main(int argc, char *argv[])
{
```

C function prototypes and structs:

```
int accept(int sock, struct sockaddr *addr, int addrlen)
int bind(int sock, struct sockaddr *addr, int addrlen)
int close(int fd)
int closedir(DIR *dir);
int connect(int sock, struct sockaddr *addr, int addrlen)
int dup2(int oldfd, int newfd)
int execl(const char *path, char *argv0, ..., (char *)0)
int execv(const char *path, 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, ...)
struct hostent *gethostbyname(const char *name)
int kill(int pid, int signo)
int listen(int sock, int n)
int open(const char *path, int oflag)
DIR *opendir(const char *name);
int pclose(FILE *stream)
int pipe(int filedes[2])
FILE *popen(char *cmdstr, char *mode)
struct dirent *readdir(DIR *dir);
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)
        /*cmd has the value SETVAL, GETVAL, IPC_RMID */
int semget(key_t key, int nsems, int semflags)
int semop(int semId, struct semops *sem_ops, int nops)
int sigaction(int signum, const struct sigaction *act, struct sigaction *oldact);
int sigaddset(sigset_t *set, int signum);
int sigemptyset(sigset_t *set);
int sigprocmask(int how, const sigset_t *set, sigset_t *oldset);
       /*how has the value SIG_BLOCK, SIG_UNBLOCK, or SIG_SETMASK */
unsigned int sleep(unsigned int seconds);
int socket(int family, int type, int protocol)
int sprintf(char *s, const char *format, ...)
int stat(const char *file_name, struct stat *buf);
char *strncat(char *dest, const char *src, size_t n);
int strncmp(const char *s1, const char *s2, size_t n);
char *strncpy(char *dest, const char *src, size_t n);
int wait(int *status)
int waitpid(int pid, int *stat, int options) /* options = 0 or WNOHANG*/
void Writen(int filedes, const void *buf, size_t nbytes);
```

```
struct sigaction {
   void (*sa_handler)(int);
   sigset_t sa_mask;
   int sa_flags; /*0*/
}

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

struct hostent {
   char *h_name; /* official name */
   char **h_aliases; /* alias list */
   int h_addrtype; /* host address type */
   int h_length; /* length of address */
   char *h_addr; /*address*/
}
```

```
WIFEXITED(status)      WEXITSTATUS(status)
WIFSIGNALED(status)    WTERMSIG(status)
WIFSTOPPED(status)     WSTOPSIG(status)
```

Shell and Perl comparison operators

Shell	Perl	Description
-d filename	-d filename	Exists as a directory
-f filename	-f filename	Exists as a regular file.
-r filename	-r filename	Exists as a readable file
-w filename	-w filename	Exists as a writable file.
-x filename	-x filename	Exists as an executable file.
-z string	string eq ""	True if empty string
str1 = str2	str1 eq str2	True if str1 equals str2
str1 != str2	str1 ne str2	True if str1 not equal to str2
int1 -eq int2	int1 == int2	True if int1 equals int2
-ne, -gt, -lt, -le	!=, >, >=, <, <=	For numbers
!=, >, >=, <, <=	ne, gt, lt, le	For strings
-a, -o	&&, ||	And, or.

Perl functions:

```
push(ARRAY, LIST)
pop(ARRAY) -- returns LIST
sort BLOCK LIST -- returns LIST
defined(SCALAR) -- returns true if SCALAR is defined, false otherwise
split(/PATTERN/, SCALAR) - returns LIST
open(FILEHANDLE, EXPR) -- return true if filename given by EXPR is opened

Meta characters that need to be escaped are \ | ( ) [ { . $ ^  ? +
```

Perl pattern matching:

\s = [\t\n\r\f]	space
\w = [a-zA-A0-9_]	word
\d = [0-9]	digit
[]	one character of a set
[^]	any expect one of the set
+	one or more
*	zero or more
?	zero or one
.	any character

Perl Special variables

@_	arguments to a subroutine
$_	the default scalar variable
$.	the current input line number
$0	program name
$$	process id
$!	last system call error
@ARGV	command line arguments

Python String Methods:

```
s.lower(), s.upper(), s.strip(), s.isdigit(),
s.find(path, start, end), s.replace(old, new, limit)
```

Python List Methods:

```
L.reverse(), L.sort, L.insert(i, x),
L.remove(x), L.append(x), L.count(x)
```

Other Python Methods:

```
min(x), max(x), len(x), x in L, L = s.split(c), s.join(L), int(s), float(s)
```

Python File I/O

```
f = open(name, mode), s = f.read(n), s = f.readline(), L = f.readlines(),
f.write(s), f.close()
sys.argv - command line arguments
```