CS252 Final Review Homework

Answer this practice exam and turn it in pdf format using the following command in data:

```
turnin -c cs252 -p hw2 hw2.pdf
```

The homework is due Wednesday May 6th, 2020 at 11:59pm EST

1. Write a program "grepsort arg1 arg2 arg3" that implements the command "grep arg1 | sort < arg2 >> arg3". The program should not return until the command finishes. "arg1", "arg2", and "arg3" are passed as arguments to the program. Example of the usage is **"grepsort hello infile outfile"**. This command will print the entries in file **infile** that contain the string **hello** and will append the output sorted to file **outfile**. Use pipes. Do error checking. Notice that the output is appended to arg3.

```
const char *usage = "Usage:\tgrepsort arg1 arg2 arg3\n\tgrep arg1 | sort  < arg2 >> arg3\n";
int main(int argc, char **argv)
{
        if (argc != 4)
        {
                fprintf(stderr, "%s\n", usage);
                exit(1);
        }

        int defaultin = dup(STDIN_FILENO);
        int defaultout = dup(STDOUT_FILENO);
        int pipefd[2];

        if (pipe(pipefd) == -1)
        {
                perror("pipe");
                exit(1);
        }

        int infd = open(argv[2], O_RDONLY);
        if (infd < 0)
        {
                perror("open in");
                exit(1);
        }

        dup2(infd, STDIN_FILENO);
```

```
close(infd);

dup2(pipefd[1], STDOUT_FILENO);
close(pipefd[1]);

int ret = fork();
//child
if (ret == 0)
{
        close(pipefd[0]);

        char *args[3];
        args[0] = "grep";
        args[1] = argv[1];
        args[2] = NULL;

        execvp(args[0], args);
        perror("execvp");
        _exit(1);
}
else if (ret < 0)
{
        perror("fork");
        exit(1);
}

int outfd = open(argv[3], O_WRONLY | O_APPEND | O_CREAT, 0666);
if (outfd < 0)
{
        perror("open out");
        exit(1);
}

dup2(outfd, STDOUT_FILENO);
close(outfd);

dup2(pipefd[0], STDIN_FILENO);
close(pipefd[0]);

ret = fork();
if (ret == 0)
{
        close(pipefd[1]);
        char *args[2];
        args[0] = "sort";
        args[1] = NULL;

        execvp(args[0], args);
        perror("execvp");
        _exit(1);
    }
    else if (ret < 0)
    {
        perror("fork");
        exit(1);
    }

    dup2(defaultin, STDIN_FILENO);
    dup2(defaultout, STDOUT_FILENO);
    close(defaultin);
    close(defaultout);

    waitpid(ret, 0, 0);

    return 0;
}
```

2. Complete the procedure runCommand*(command, outputBuffer, bufferSize)* that executes a command in a different process and stores its output in outputBuffer. *command* is the name of the program with no arguments. See how main uses *runCommand*().*runCommand* will return 0 on success or -1 otherwise. Use a pipe to communicate the parent and the child process running runCommand(). Have the parent read from the pipe and write into the outputBuffer.

```
int
runCommand( char * command, char * outputBuffer, int maxBufferSize)
{

int tmpin = dup(0);
int tmpout = dup(1);
int pipe[2];
pipe(pipe);

dup2(pipe[1], tmpout);

int ret = fork();
if(ret == 0){

char args[2];
args[0] = command;
args[1] = NULL;
execvp(args[0], args);
perror("execvp");
return -1;

}

dup2(outputBuffer, tmpout);

dup2(tmpin,0);
dup2(tmpout,1);

waitpid(ret, NULL);

return 0;

}
int
main()
{
        // The output of "ls" will be stored in buffer
        char buffer[ 1024 ];
        if ( runCommand( "ls", buffer, 1024 ) < 0 ) {
        perror("runCommand" );
        exit( -1 );
        }
        printf( "ls: %s\n", buffer );
        exit( 0 );
}
```

3. Add the necessary code to the insert() and removeFirst() functions to make them synchronized. removeFirst() will have to wait if the list is empty. insert() will have to wait if there are already 20 elements in the list. **Use semaphores**. Add also the variables you need.

```
struct List {
  int val;
  int next;
};
struct List * head = NULL;
// More variables

main()
{
 // DO any initializations here

pthread_mutex_init(&mutex, NULL);
sema_init(&emptySem, 0, ...);
}
void insert( int val )
{

  List tmp = new List;
pthread_mutex_lock(&mutex);
  tmp->val = val;

  tmp->next = head;

  head = tmp;
pthread_mutex_unlock(&mutex);
sema_post(&emptySem);
}
Struct List * removeFirst()
{
```

```
sema_wait(&emptySem);
pthread_mutex_lock(&mutex);
  List tmp = head;

  head = tmp->next;
pthread_mutex_unlock(&mutex);
  return tmp;
}
```

4. Using C++ and Semaphores write a class SynchronizedStackSemaphores of int values where pop() will block if the stack is empty and push will block if the stack is full. Write the member variables that you think are necessary. Implement the stack with an array of int's and allocate it dynamically in the constructor. Hint: Use the "Bounded Buffer Problem" with semaphores as an example in your implementation.

```
#include <synch.h>
#include <pthread.h>

class SynchronizedStackSemaphores {
  // Add your member variables here
  int top;
  int * stack;

  public:
    SynchronizedStackSemaphores(int maxStackSize);
    void push(int val);
    int pop();
};

SynchronizedStackSemaphores::SynchronizedStackSemaphores(int maxStackSize) {
    top = 0;
    stack = new int[maxStackSize];



}

void SynchronizedStackSemaphores::push(int val) {




}
```

```
int SynchronizedStackSemaphores::pop(){




}
```

5. From lab3, assuming you have a procedure *void dispatchHTTP(int slaveSocket)* that processes the request and closes *slaveSocket,* write the loop server code for a) iterative server, b) concurrent server using fork, c) concurrent server creating a thread after each request, and d) pool of threads, in the procedures indicated. Each procedure receives as argument the master socket already initialized and ready to be used inside accept.

```
void iterativeServer( int masterSocket) {

int slaveSocket = accept(masterSocket, &sockInfo, &alen);
        if(slaveSocket >= 0){
                dispathHttp(slaveSocket);
        }



}
void forkServer( int masterSocket) {

int slaveSocket = accept(masterSocket, &sockInfo, &alen);
if(slaveSocket >= 0){
ret = fork();
if(ret == 0){
dispatchHttp(slaveSocket);
exit(0);
}
close(slaveSocket);
}





}
```

```
void poolOfThreads( int masterSocket) {

for(int i = 0; i < num_of_threads; i++){
pthread_create(&thread[i], NULL, loopthread, masterSocket);
}
loopthread(masterSocket);

}

void createThreadForEachRequest( int masterSocket ) {


int slaveSocket = accept(masterSocket, &socketInfo, &alen);
if(slaveSocket >= 0){
pthread_create(&thread, NULL, dispathHttp, slaveSocket);
}


}

// Other procedures
void* loop_thread(int masterSocket){
int slaveSocket = accept(masterSocket, &socketInfo, &alen);
if(slaveSocket >= 0){
dispathHttp(masterSocket);
}
```

6. Implement a R/W lock class.

RWLock.h

```
class RWLock
{
   int _nreaders;

     sema_t _semAccess; mutex_t _mutex;
public:
    RWLock();
    void readLock();
    void writeLock();
    void readUnlock();
    void writeUnlock();
};
```

RWLock.cpp

```
RWLock::RWLock(){
int _nreaders = 0;
sema_init(semAccess, 1, ..);
pthread_mutex_init(mutex, NULL);
}
void RWLock::readLock(){
pthread_mutex_lock(&mutex)
_nreaders++;
if(_nreaders == 1){
sema_wait(&semAccess);

}
pthread_mutex_unlock(&mutex);
}


void RWLock::readUnlock(){
pthread_mutex_lock(&mutex);
_nreaders--;
if(_nreaders == 0){
sema_post(&semAccess);
}
pthread_mutex_unlock(&mutex);

}
```

```
void RWLock::writeLock(){
sema_wait(&semAccess);
}

void RWLock::writeUnlock(){
sema_post(&semAccess);
}
```

7. What are the four parameters that a computer needs to be able to get connected to the internet and what are they used for?

Local IP address – Your current address
Subnet Mask – It is used to send packets to hosts in the same LAN
Default Router – It is used to send packets to the Internet
Default DNS Server - It converts the name to IP addresses

8. How does a computer know when it can deliver a packet directly and when it has to pass a packet to a router?

Do a bitwise operation using subnet mask on the ip destination address. If the resulting ip address is the same as the prefix local ip address, then send the packet locally or else send it over to a router

9. What does ARP mean and how does it work?
ARP stands for Address Resolution Protocol. When you send a packet over from one computer to another over the internet. The destination ip address has to be translated to an ethernet address of the destination machine.

There is ARP input where the IP address A in locally connected network N and ARP output which is the Ethernet address for A. The ARP keeps binding between IP address and Ethernet address in an ARP cache table

10. What does DNS mean and what it is used for?
DNS stands for Domain Name Server. It is used to translate the domain name to an ip address

11. What does DHCP mean and how does it work?
DHCP stands for Dynamic Host Configuration Protocol. It makes so that the internet configuration automatically prevents conflicts. It also configures the local IP address, subnet mask, default router, and DNS server. Additionally there is no need for the administrator to manually configure it

12. What does UDP mean?
UDP stands for User Datagram Protocol.
It is message oriented, unreliable and has minimal overhead.

13. What does TCP mean? What are the 6 features of TCP?
TCP stands for Transmission Control Protocol.
1) It is reliable for example the acknowledgement and retransmission ensures that the packet reaches
2) The retransmission makes it so that the sender starts timer, if it did not receive the acknowledgement when the message is transmitted, it will expire and then send again
3) The acknowledgement where the receiver has to know when the data arrives
4) It is connection oriented which requires a connection both ways
5) It is similar to a Read Write File
6) It is full duplex where the communication happens both ways

14. When should you use TCP and when should you use UDP?
For TCP:
- When reliability is a concerned and making sure the packet is sent to the correct destination
- When speed doesn’t matter as TCP usually may be slow.
- For example when connecting to a game server

For UDP:
- It is unreliable but fast so when speed matters a lot.
- The packets may be lost or rough on the way to destination, but it has a minimal delay
- For example this is best with a Voice over Internet Protocol

15. What does NAT stand for? Assume that a packet <A, 4563, X, 80> is sent from a host behind a NAT box to a webserver X. Describe the steps for the translation (6 steps) since it goes from the host A, through the NAT box, to X and then back from X to the NAT box to A.

NAT stands for Network Address Translation.

1) The Packet <A, 4563, X 80> needs to be sent from computer A to computer X via the internet
2) Then the Packet get sends to the NAT box, NAT box converts port 4563 into a random unused port lets day 1234. It also converts the ip source address to a NAT box default ip address and gets stored in a NAT table
3) <NAT, 1234, X, 80> packet gets send over to destination X
4) When the packet arrives from X, eg. <X, 80, NAT, 1234> to the NAT box
5) NAT Box handles mapping conversion, maps port 1234 to original port 4563, NAT box transfer packet to computer

16. Explain why NAT boxes can be used as firewalls to prevent unwanted connections. Also explain why it is not normally possible to run web servers behind a firewall and how this problem can be solved.

- Packets only get sent over to a computer by the NAT box and only if it requests for one
- Any other packets arrived to the NAT box that are not listed in the NAT table get discarded

Run Webservers
- The webserver, will be expecting the computer to have requests be made to the server.
- By doing so, the requests will come by the NAT box, and since those requests could not be mapped to the NAT table, the packets get discarded

How to solve it:
- A proxy server that connects directly between the requester and the server
- To manually configure the NAT box to do a port forwarding

17. Write a simple client program "echo-client host port string" that sends a string "string" followed by "\r\n"to "host : port" and then it reads the server's response and prints it to stdout.

```
int main(int argc, char** argv){
if (argc < 4){
        return -1;
}

char *host = argv[1];
int port = argv[2];
char *str = argv[3];

struct sockraddr_in socketAddress;
memset((char*) &socketAddress, 0 ,sizeof(socketAddress));
socketAddress.sin_family = AF_INET;

if (port > 0){
socketAddress.sin_port = htons(port);
}

struct hostent *ptrh = gethostbyname(host);

memcpy(&socketAddress.sin_addr, ptrh->h_addr, ptrh->h_length)

struct protoent *ptrp = getprotobyname("tcp");

int sock = socket(PF_INET, SOCK_STREAM, ptrp->p_proto);

int read = read(sock, string, 100);
string[read] = 0;
printf("string=%s\n", string);

write(sock, string, strlen(string))
write(sock, "\r\n", 2);


char buf[1024];
int n = recv(sock, buf, sizeof(buf), 0);
while(n > 0){
        write(1,buf, n)
        n = recv(sock, buf, sizeof(buf), 0);
        }


}
```

18. Write a simple iterative server "echo-server port" that waits for incoming requests in "port" and once it receives a string delimited by "\r\n" it will reply with the same string plus "\r\n" and close the connection.

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

if(argc < 2){
        exit(1);
}

int port = argv[1];

struct sockaddr_in serverIPAddress;
memset( &serverIPAddress, 0, sizeof(serverIPAddress));
serverIPAddress.sin_family = AF_INET;
serverIPAddress.sin_addr.s_addr = INADDR_ANY;
serverIPAddress.sin_port = htons((u_short) port);

int masterSocket = socket(PF_INET, SOCK_STREAM, 0);

int err bind(masterSocket, (struct sockaddr*)&serverIPAddress, sizeof(serverIPAddress));

err = listen(masterSocket, length_queue);

while(1){
        int slaveSocket = accept(masterSocket, &sockInfo, &alen);
        if (slaveSocket >= 0){
        dispatchHttp(slaveSockt);

        }
}
```

19. Enumerate 5 of the 12 questions in "Joel's Test".
1) Do you use a source control?
2) Do you use a bug tracker?
3) Do you use the best tool money can buy?
4) Do you write test/spec?
5) Do you fix bugs before writing a new feature?

20. What is XP programming?
XP stands for Extreme Programming. You first write the test then code and then launch in iteration.

21. From XP Programming, mention 4 items from the Planning List, 4 Items from the Coding List, 4 Items from the Designing List, and 4 Items from the testing List.

Planning
- User stories
- Stand up meeting
- Estimate build time
- Iteration

Coding
- Write Test first
- Write code to pass the test
- Coding standard
- Pair programming

Designing
- Class design
- Simple
- CRC cards
- Metaphore

Testing
- Unit test for each class
- Code must pass integration test
- Do an acceptance test when the bug is found
- Create acceptance tests

22. Explain 5 uses of the source control system.

- Make collaboration easy
- Helps resolve conflict
- Go back in time and retrieve old codes
- Open source code via distributed source control
- Helps merging

23. Describe the advantages and disadvantages of centralized vs. distributed source control systems.

Central
Pros: Good for large corporation where everyone is working on one repository
Cons: Repository has to be healthy, as a bad commit will stop other team members from working

Distributed
Pros: Good for open source such as Github and programmers can work individually without messing with central repository
Cons: More complex than central repository

24. Describe the 4 types of tests, who writes these tests in the organization, and when do they run.

Unit Testing: test each class individually, written by the programmer
Acceptance Test: quality test before release, written by the Quality Assurance
System Test: test a specific subsystem, multiple classes, written by the programmer and the QA
Regression Test: after a bug is fixed, run the same test again to verify it, written by the programmers

25. Explain why it is important to have a bug tracking system.

- We can see a list of features and bugs to implement and fix
- We can assign a specific features/bugs to a team member
- We can track the progress and comments
- We can assign a level of severity and priority to a bug

26. Explain the difference between Priority and Severity in a bug.

Priority means how it is important to the team, for example making the code cleaner
Severity means that if it is severe, it prevents users from using the application

27. Mention 5 cases when you can apply refactoring.

1) When the method parameters are too long
2) A method is trying to do too much
3) Comments

4) Coupling of classes
5) Duplicated code

28. What is a Software Pattern, what are the parts of a software pattern? What is the name of the book that introduced software patterns and the authors?

Software Pattern means the proven design patterns that solves a generic problem
These patterns are:
- synopsis
- context
- forces
- solutions
- consequences
- implementation

The book that introduced this is:Design Pattern: Elements of Reusable OO Software by Gamma

29. Describe the Proxy Pattern and 2 applications.

A Proxy Pattern is used to solve a problem where there is a need of an object to be illusively local when it is remote. This will allow the proxy object to be extended orthogonally.
Proxys and service profiling objects can inherit from a common class/interface

30. Describe the Command Pattern and two applications.
Command Pattern is used to encapsulate commands in objects to control selection and sequencing, queue, etc..
We can put all the command management and sequence in an interface. The command object can implement that interface and a command manager tracks the sequence of queue

31. What is the difference between Code Instrumentation Profiling and Statistical Sampling Profiling?

Code Instrumentation Profiling is used for writing code before and after the method to calculate method run time

Statistical Sampling Profiling is used to run the software that calculates average run time of the application

32. Explain why Optimizing should be left until the very end in the software cycle and why you should use an execution profiler before attempting to optimize a program.

Left at the end:
- not everything in the code must be optimized
- optimization may cause the code to be unreadable

execution profiler:
- to detect which part of the code really requires optimization

33. Assume the following table called "customers":

CompanyName	**ContactName**	**Address**	**City**
Alfreds Futterkiste	Maria Anders	Obere Str. 57	Berlin
Berglunds snabbköp	Christina Berglund	Berguvsvägen 8	Luleå
Centro comercial Moctezuma	Francisco Chang	Sierras de Granada 9993	México D.F.
Ernst Handel	Roland Mendel	Kirchgasse 6	Graz
FISSA Fabrica Inter. Salchichas S.A.	Diego Roel	C/ Moralzarzal, 86	Madrid
Galería del gastrónomo	Eduardo Saavedra	Rambla de Cataluña, 23	Barcelona
Island Trading	Helen Bennett	Garden House Crowther Way	Cowes
Königlich Essen	Philip Cramer	Maubelstr. 90	Brandenburg
Laughing Bacchus Wine Cellars	Yoshi Tannamuri	1900 Oak St.	Vancouver
Magazzini Alimentari Riuniti	Giovanni Rovelli	Via Ludovico il Moro 22	Bergamo
North/South	Simon Crowther	South House 300 Queensbridge	London
Paris spécialités	Marie Bertrand	265, boulevard Charonne	Paris
Rattlesnake Canyon Grocery	Paula Wilson	2817 Milton Dr.	Albuquerque
Simons bistro	Jytte Petersen	Vinbæltet 34	København
The Big Cheese	Liz Nixon	89 Jefferson Way Suite 2	Portland
Vaffeljernet	Palle Ibsen	Smagsløget 45	Århus
Wolski Zajazd	Zbyszek Piestrzeniewicz	ul. Filtrowa 68	Warszawa

Write the result of the following queries (You can use a description when the number of rows in the resulting table is larger than 5, otherwise write down the whole resulting table).

a) SELECT *FROM customers

Will return all information from the customers

b) SELECT ContactName FROM customers

   Returns all ContactName of all customers

c) SELECT CompanyName FROM customers WHERE ContactName LIKE Liz%

   Returns the Big Cheese

d) SELECT CompanyName, ContactName FROM customers WHERE City LIKE Portland

   Returns the Big Cheese and Liz Nixon

e) Write a query to get the companies that are in Spain

   SELECT CompanyName FROM customers WHERE City LIKE "Spain"

f) Write a query to get all the companies that start with R or W

   SELECT * FROM customers WHERE CompanyName LIKE "R%" OR CompanyName LIKE "R%"