CMPSCI 377 Exam (Sample)

You will have 90 minutes to work on this exam, which is closed book.

Please write your name on EVERY page.

You are to abide by the University of Massachusetts Academic Honesty Policy.

Name:_____

Problem 1	_____ out of 20
Problem 2	_____ out of 20
Problem 3	_____ out of 15
Problem 4	_____ out of 20
Problem 5	_____ out of 25
Total	**_____ out of 100**

1. Problem 1: Multi-level page tables (approx. 15 minutes)

Consider a computer with the following properties:

- 32-bit addresses, byte addressing
- 256-byte pages
- A page table entry occupies 4 bytes
- No TLB (translation lookaside buffer)

The machine can use 1-level page tables, or 3-level paging (with 8 address bits devoted to each level).

a. For 1-level paging, if all addresses are in use, how many page tables are needed and how much memory is occupied by page tables?

b. For 3-level paging, if all addresses are in use, how many page tables are needed and how much memory is occupied by page tables at each level?

c. For 1-level paging, if only the lowest 4,096 and highest 4,096 addresses are in use, how many page tables are needed and how much memory is occupied by page tables? (Recall that 4,096 equals 2^{12})

d. For 3-level paging, if only the lowest 4,096 and highest 4,096 addresses are in use, how many page tables are needed and how much memory is occupied by page tables at each level?

e. Discuss the relative advantages and disadvantages of 1-level paging and 3-level paging, assuming all addresses are used.

f. Discuss the relative advantages and disadvantages of 1-level paging and 3-level paging, assuming only the lowest 4,096 and highest 4,096 addresses are used.

2. Lazy fork (approx. 18 minutes)

As discussed in class, Unix `fork` initializes the contents of the child's address space by copying the contents of the parent's address space when fork is called. Your job is to speed up `fork` by deferring or avoiding the work of copying data between the two address spaces. Your solution should reduce the number of bytes copied as much as possible for the two common ways of using `fork`: running two copies of the same program, and starting a new program by using `exec` after `fork`. Other than increased speed, user processes should not notice any difference between the old and new `fork`. Assume that address spaces are represented by a simple page table, and that the MMU respects read/write protection bits and sets reference and dirty bits appropriately. Hint: think about sharing data between the parent and child. Describe how to modify fork to defer or avoid copying. Be specific and concise in your answers.

a. What is the unit of copying, and why should/can the copy not be done on a larger or smaller unit?

b. What events in the parent will trigger a copy? What events in the child will trigger a copy?

c. Describe how the page tables and MMU data are manipulated and used when `fork` is called.

d. Describe how the page tables and MMU data are manipulated and used when a copy must be made.

3. Communications protocols for Project 3 (approx. 12 minutes)

This question asks about different aspects of the communication protocol used in Project 3. For your reference, the page following this question contains the sections from Project 3 that describe the communication protocol (for FS_APPEND requests, as an example). Project 3 used TCP sockets to communicate between the client and server. You have been asked to change the communication protocol to use UDP sockets instead. Assume that a pair of UDP sockets (one on the client, one on the file server) are set up for each client, and that this pair of sockets is used for ALL requests made by that client (this differs from how Project 3 used TCP, which set up a new pair of sockets for each REQUEST). Assume that UDP messages are unlimited in size.

a. The original cleartext request header includes a size field. Is this size field necessary when using UDP? Why or why not?

b. TCP provides several abstractions that UDP does not. In particular, TCP provides four abstractions that protect the receiver from seeing certain changes to a sequence of network messages. State four changes to a message or sequence of messages that are visible to UDP receivers, but not to TCP receivers (this question is independent of the Project 3 communication protocol).

change 1:

change 2:

change 3:

change 4:

c. The existing communication protocol includes a sequence number in the request message. Which one of the changes to messages in part b does this help the client and file server to handle (if you think there are several, pick the change for which sequence numbers helps the most)? Justify your answer.

d. The existing communication protocol encrypts the request message. Which one of the changes to messages in part b does this help the client and file server to handle (if you think there are several, pick the change for which encrypting the request helps the most)? Justify your answer.

3.3 FS_APPEND

A client appends to an existing file by sending an FS_APPEND request to the file server. An FS_APPEND request message is a string of the following format:

"FS_APPEND <username> <session> <sequence> <filename> <size><NULL>"<data>

<username> is the name of the user making the request
<session> is the session number for this request
<sequence> is the sequence number for this request
<filename> is the name of the file to which the data is being appended
<size> specifies the number of bytes to append to the file
<NULL> is the ascii character '\0' (terminating the string)
<data> is that data to append to the file. Note that <data> is outside of the request string (i.e. after <NULL>). The size of <data> is given in <size>.

Upon receiving an FS_APPEND request, the file server should check the validity of its parameters. The server should also check that that the file exists, is owned by <username>, and that there is sufficient disk space to satisfy the request.

If the request is allowed, the file server should append the data to the file (writing to disk in the order the bytes appear in the file). The response message for a successful FS_APPEND follows the following format: "<session> <sequence><NULL>"

<session> is the session number from the request message
<sequence> is the sequence from the request message
<NULL> is the ascii character '\0' (terminating the string)

No data should be appended to the file for unsuccessful requests.

4 Encryption

Each of the request messages described in Section 3 will be encrypted using the password parameter that was passed to the client function. To enable the file server to decrypt the request message, the client will send a cleartext (i.e. un-encrypted) request header before sending the request message. The cleartext request header follows the following format:

"<username> <size><NULL>"

<username> is the name of the user that was passed to the client function. The file server uses this information to choose which password to use to decrypt the ensuing request message.
<size> is the size of the encrypted message that follows this cleartext request header
<NULL> is the ascii character '\0' (terminating the string)

Each response message from the file server will be encrypted using the user's password. To enable the client to receive and decrypt the response message, the file server will send a cleartext response header before sending the response message. The cleartext response header follows the following format:

"<size><NULL>"

<size> is the size of the encrypted message that follows this cleartext response header
<NULL> is the ascii character '\0' (terminating the string)

4. Unix filesystems (approx. 15 minutes)

Consider a filesystem with the following characteristics:

- Hierarchical directories are supported.
- Directories and files use the same storage structure on disk (specified below). A flag in the inode identifies whether the inode belongs to a file or to a directory.
- The data blocks in a directory contain (filename, disk-block pointer) tuples.
- The size of a disk block is 8192 bytes.
- Disk block pointers are 4-byte unsigned integers.

A non-uniform depth, multi-level indexed scheme is used, where the inode contains 15 disk block pointers as follows:

- 12 pointers directly to data blocks
- 1 pointer to a single indirect block (a single indirect block is filled with pointers to data blocks)
- 1 pointer to a double indirect block (a double indirect block is filled with pointers to single indirect blocks)
- 1 pointer to a triple indirect block (a triple indirect block is filled with pointers to double indirect blocks)

a. Assume the location of the root inode on disk is known to the kernel, but no inodes or other filesystem information is cached in memory. Also assume that all directories happen to be small enough to fit in just one data block. The file /etc/bigfile is 128 kilobytes. To read the only the last byte of it, how many disk accesses are necessary? Give a list that explains the function of each of these disk accesses.

b. Assuming the results of all the previous accesses are cached in memory, how many disk accesses, if any, are needed to read one byte from the middle (offset 64 KB) of the same file?

c. What is the smallest file size (in bytes) for which it would be necessary to use the triple indirect block? (An approximate answer accurate to within 10% is acceptable, and it is acceptable to state your answer as an expression, for example 2 raised to some power.)

d. What is the maximum size of a filesystem with these characteristics?

e.Would it ever be useful to add a quadruple indirect block to this filesystem (assuming absolutely nothing else is changed)? Briefly explain why or why not.

5. Secure communication (approx. 20 minutes)

Alice and Bob want to use public-key encryption to establish a symmetric key K, then use that symmetric key K to encrypt further communication. Your job is to analyze each of the following protocols and explain how a malicious third party (Eve) can read messages encrypted with K. Assumptions:

- All parties know the correct public key for each other party; each private key is known only by its owner.
- Eve, a malicious third-party, can read all network traffic and can insert her own messages, but can not modify or delete messages.
- Any party can initiate a key exchange protocol with Bob, and Bob will respond as described.
- encrypt() == decrypt(), so the term crypt() is used for both in the protocols described below

a. Protocol 1

Alice constructs a symmetric key, K and sends it to Bob:
`"Alice to Bob" "The key is" K`

Bob returns K to Alice:
`"Bob to Alice" "The key is" K`

What steps should Eve take in order to read subsequent messages between Alice and Bob encrypted with K?

b. Protocol 2

Alice constructs a symmetric key K, encrypts it with her private key, and sends the result to Bob:
`"Alice to Bob" crypt("The key is" K, Alice-private)`

Bob recovers K by decrypting the message with Alice's public key. Bob then encrypts K with his private key, and sends the result to Alice: `"Bob to Alice" crypt("The key is" K, Bob-private)`

What steps should Eve take in order to read subsequent messages between Alice and Bob encrypted with K?

c. Protocol 3

Alice constructs a symmetric key K, encrypts it with Bob's public key, encrypts the result with her private key, and sends the result to Bob.

```
"Alice to Bob" crypt(crypt("The key is" K, Bob-public),
Alice-private)
```

Receiving this message, Bob encrypts it with Alice's public key, then recovers K with his private key, then encrypts and signs K and sends the result to Alice.

```
"Bob to Alice" crypt(crypt("The key is" K, Alice-public),
Bob-private)
```

What steps should Eve take in order to read subsequent messages between Alice and Bob encrypted with K?