Midterm Exam #2 Nov 3, 2016

COMS W3157 Advanced Programming Prof. Jae Woo Lee
Fall 2016 Columbia University

About this exam:

- There are 3 problems totaling 100 points:

  Problem 1: 21 points
  Problem 2: 49 points
  Problem 3: 30 points

- Assume the following programming environment:

  All programs are built and run on Ubuntu Linux 16.04, 64-bit version, where sizeof(int) is 4 and sizeof(int *) is 8.

  All library function calls and system calls are successful. For example, you can assume malloc() does not return NULL.

  For all program code in this exam, assume that all the necessary #include statements are there even if they are not shown.

  If this exam refers to lab code, assume the versions provided by Jae, i.e., skeleton code and solutions.

  When writing code, avoid using hardcoded numbers as much as possible. Hardcoded numbers make your program error prone, less extensible, and less portable. For example, using "sizeof(int *)" instead of "8" will make it correct for both 32-bit and 64-bit environments.

What to hand in and what to keep:

- At the end of the exam, you will hand in only the answer sheet, which is the last two pages (one sheet printed double-sided) of this exam booklet. You keep the rest of the exam booklet.

- Make sure you write your name & UNI on the answer sheet.

- Please write only your final answers on the answer sheet. Verbosity will only hurt your grade because, if we find multiple answers to a question, we will cherry-pick the part that will result in the LOWEST grade. This policy ensures that a shotgun approach to solving a problem is never rewarded. Please make sure you cross out clearly anything that you don’t consider your final answer.

- Before you hand in your answer sheet, please copy down your answers back onto the exam booklet so that you can verify your grade when the solution is published in the mailing list.

Good luck!

```
+-------------------------------------------------------------------+
|  PLEASE DO NOT OPEN THIS EXAM BOOKLET UNTIL YOU ARE TOLD TO DO SO!  |
+-------------------------------------------------------------------+
```

References

FILE *fopen(const char *filename, const char *mode)

  - Opens a file, and returns a FILE* that you can pass to other
    file-related functions to tell them on which file they should operate.

  - "r" open for reading (file must already exist)
    "w" open for writing (will trash existing file)
    "a" open for appending (writes will always go to the end of file)

    "r+" open for reading & writing (file must already exist)
    "w+" open for reading & writing (will trash existing file)
    "a+" open for reading & appending (writes will go to end of file)

  - returns NULL if file could not be opened for some reason

int fseek(FILE *file, long offset, int whence)

  - Sets the file position for next read or write. The new position,
    measured in bytes, is obtained by adding offset bytes to the position
    specified by whence. If whence is set to SEEK_SET, SEEK_CUR, or
    SEEK_END, the offset is relative to the start of the file, the current
    position indicator, or end-of-file, respectively.

  - returns 0 on success, non-zero on error

size_t fread(void *p, size_t size, size_t n, FILE *file)

  - reads n objects, each size bytes long, from file into the memory
    location pointed to by p.

  - returns the number of objects successfully read, which may be less than
    the requested number n, in which case feof() and ferror() can be used to
    determine status.

size_t fwrite(const void *p, size_t size, size_t n, FILE *file)

  - writes n objects, each size bytes long, from the memory location pointed
    to by p out to file.

  - returns the number of objects successfully written, which will be less
    than n when there is an error.

char *fgets(char *buffer, int size, FILE *file)

  - reads at most size-1 characters into buffer, stopping if newline is read
    (the newline is included in the characters read), and terminating the
    buffer with '\0'

  - returns NULL on EOF or error (you can call ferror() afterwards to find
    out if there was an error).

int fputs(const char *str, FILE *file)

  - writes str to file.
  - returns EOF on error.

Problem [1]: 21 points total, 7 parts, 3 points each

Please write "True" or "False" for the true/false questions, and write down exactly ONE choice for the multiple-choice questions, on the answer sheet.

(1.1) Recall the five Internet protocol layers that we discussed in class. Which of the following is an application layer (L5) protocol?

(A) HTTP
(B) TCP
(C) IP
(D) Ethernet
(E) All of the above
(F) None of the above

(1.2) Which of the following Sockets API functions is NOT called by the TCPEchoServer.c program?

(A) socket()
(B) connect()
(C) bind()
(D) listen()
(E) accept()
(F) TCPEchoServer.c program calls ALL of the functions listed above
(G) TCPEchoServer.c program calls NONE of the functions listed above

---

For (1.3) - (1.7), consider the following netcat process currently executing in a pipeline:

```
cat - | nc -l 44444 | cat -
```

Assume that a TCP connection between this netcat process and a remote netcat process has been established.

Note that the "cat" command will read bytes from standard input when you give "-" as an argument instead of a file name.

(1.3) This netcat process is currently running in which mode?

(A) Server mode
(B) Client mode
(C) Both server and client mode
(D) Neither server nor client mode

(1.4) This netcat process will copy bytes it receives from the TCP connection back out to the same TCP connection. (True/False)

(1.5) This netcat process will always receive from the TCP connection the same bytes that the process has previously sent out to the same TCP connection. (True/False)

(1.6) The bytes that the netcat process receives from the standard input will be sent to the remote netcat process. (True/False)

(1.7) The bytes that the netcat process receives from the TCP connection will be written out to its standard output. (True/False)

Problem [2] (49 points): Consider the following shell session on CLAC:

```
jae@clac ~/tmp $ cp /home/jae/cs3157-pub/bin/mdb-cs3157 .
jae@clac ~/tmp $ cp /home/jae/cs3157-pub/bin/mdb-lookup .
jae@clac ~/tmp $ gcc mdb-select.c -o mdb-select
jae@clac ~/tmp $ ls -l
total 208
-rw-r--r-- 1 jae jae 168360 Nov  2 13:25 mdb-cs3157
-rwxr-xr-x 1 jae jae  22247 Nov  2 13:25 mdb-lookup
-rwxr-xr-x 1 jae jae   9152 Nov  2 13:25 mdb-select
-rw-r--r-- 1 jae jae   1105 Nov  2 13:05 mdb-select.c

jae@clac ~/tmp $ ./mdb-lookup mdb-cs3157
lookup: AP
 375: {Kanjae} said {I miss the old AP}
2328: {AP student} said {I'm lonely}
2407: {confused AP stu} said {but how did you do that}
3238: {Mark} said {Hello AP!}
3246: {A} said {Hey AP!!}
3248: {Paul} said {Hello dudes of AP!}
3276: {Jill Shah} said {hello AP!}
3280: {Nick} said {Hello AP'ers}
3297: {Melanie} said {Hello AP!}
3306: {Chi} said {Good luck APers!}
3345: {Donald} said {Make AP great again}
3391: {m m} said {Thank u APr fantastic!}
3397: {tranquillo} said {death by AP [C [C [C [C}
3410: {AP} said {your gpa belong   to me}
3467: {AP student} said {Just started Lab 4}
3491: {Darren Hakimi} said {AP with Jae is great}
4004: {RandomStudent} said {hello AP!}
4100: {william} said {hello AP}
4123: {Jordan} said {I'm not in AP yet}

lookup: ^C
jae@clac ~/tmp $ ./mdb-select mdb-cs3157 mdb-fun 2328 2407 3345 3410

jae@clac ~/tmp $ ./mdb-lookup mdb-fun
lookup:
   1: {AP student} said {I'm lonely}
   2: {confused AP stu} said {but how did you do that}
   3: {Donald} said {Make AP great again}
   4: {AP} said {your gpa belong   to me}

lookup: ^C
jae@clac ~/tmp $ ./mdb-select mdb-cs3157 mdb-fun 375 3397

jae@clac ~/tmp $ ./mdb-lookup mdb-fun
lookup:
   1: {AP student} said {I'm lonely}
   2: {confused AP stu} said {but how did you do that}
   3: {Donald} said {Make AP great again}
   4: {AP} said {your gpa belong   to me}
   5: {Kanjae} said {I miss the old AP}
   6: {tranquillo} said {death by AP [C [C [C [C}

lookup: ^C
```

Problem [2] continued

The shell session demonstrates the mdb-select program, which allows you to pick out your favorite records from one mdb database file, and add them to another mdb database file.

Here are a few things you can tell from the shell session about mdb-select:

- Command line arguments to mdb-select are input mdb database file, output mdb database file, and a sequence of mdb record numbers.

- If the output database file already exists, mdb-select will append the records at the end of the file.

- If the output database file does not exist, mdb-select will create it.

- Note that the mdb-lookup program prints record numbers starting from 1 -- that is, the very first mdb record in the database file has the record number 1, not 0.

Write mdb-select.c on the answer sheet.

- You may find the following library function useful:

```
int atoi(const char *str);
```

  The man page says, "The atoi() function converts the initial portion of the string pointed to by str to int representation."

- For brevity, please do not write any error checking code in this exam. Assume that the input database file exists, the given record numbers are all valid, and all library function calls will succeed.

- Write as little code as you can. Unnecessary code will lose points.

- Do NOT declare any new variables other than the ones that are already declared in the skeleton code on the answer sheet.

- Please take care to get the syntax right. At this point, I expect that you can write correct C code without compiler. Incorrect syntax will lose points.

Problem [3]: 30 points

(3.1) Write the output of the following C program on the answer sheet:

```
        void print_char(char c)  { fwrite(&c, 1, 1, stderr); }

        static int i = 3;

        void fffork(char **a)
        {
            if (*a) {
                pid_t pid = fork();
                if (pid == 0) {
                    // child process
                    i--;
                    fffork(a + 1);
                    print_char(a[0][i]);
                } else {
                    // parent process
                    waitpid(pid, NULL, 0); // no status & no options
                    print_char('0');
                    print_char('\n');
                }
            }
        }

        int main()
        {
            char *a[4] = { "123",
                           "456",
                           "789",
                           NULL };
            fffork(a);
            return 0;
        }
```

(3.2) Draw a diagram of all fffork processes, at the point when the first character is printed on screen.

- Draw your diagram using the same style as the "f" option of the "ps" command. (Recall that you ran "ps ajxfww" command as part of lab5.) The ps man page describes the f option as follows:

        f   ASCII art process hierarchy (forest).

- Label each process with process IDs starting from 2000 -- i.e., assume that the first fffork process's process ID is 2000, and that the subsequent fffork processes are the only new processes created in the system afterwards.

- Indicate the values of the variable i, and what the pointer a is pointing to, for each process, at that moment.

- Your diagram will be evaluated not only on the information it contains, but also on how clearly it is presented. And please remember my grading policy on shotgun approaches. If you put conflicting information, the worst part of it will determine your grade.

(1.1) L5:	(1.2) API:	(1.3) mode:

(1.4) T/F:	(1.5) T/F:	(1.6) T/F:	(1.7) T/F

Problem [2]: Implement mdb-select.c

```
struct MdbRec { char name[16]; char msg[24]; };

static void die(const char *msg) { perror(msg); exit(1); }

int main(int argc, char **argv) {
    if (argc < 3) {
        fprintf(stderr,
            "usage: %s <in-mdb> <out-mdb> [mdb record numbers]\n", argv[0]);
        exit(1);
    }

    FILE *in_mdb;   FILE *out_mdb;   int mdb_rec_num;   struct MdbRec R;

    // Do NOT declare any additional variables.

    // Be succinct.  Please omit all error-checking code.
```

COMS W3157 Fall 2016
Midterm #2
Answer Sheet, page 2 of 2

Your UNI: +---+---+---+---+---+---+---+---+
| | | | | | | | |
+---+---+---+---+---+---+---+---+

Your Name: _____

left->UNI: _____ right->UNI: _____

Problem [3]
(3.1) Write the output:

_____

(3.2)