CS 162 Fall 2021

Operating Systems and Systems Programming

MIDTERM 3

INSTRUCTIONS

This is your exam. Complete it either at exam.cs61a.org or, if that doesn't work, by emailing course staff with your solutions before the exam deadline.

This exam is intended for the student with email address `<EMAILADDRESS>`. If this is not your email address, notify course staff immediately, as each exam is different. Do not distribute this exam PDF even after the exam ends, as some students may be taking the exam in a different time zone.

For questions with **circular bubbles**, you should select exactly *one* choice.

- ◯ You must choose either this option
- ◯ Or this one, but not both!

For questions with **square checkboxes**, you may select *multiple* choices.

- ☐ You could select this choice.
- ☐ You could select this one too!

You may start your exam now. Your exam is due at <DEADLINE> Pacific Time. Go to the next page to begin.

Preliminaries

(a) Full name

(b) Student ID number

(c) Autograder login (e.g. `student123`)

1. (21.0 points) True or False

Please pick whether each statement is true or false and explain why. Explanations must be two sentences or less. You may not use semicolons to circumvent this limit. Any answer longer than two sentences will receive a 0.

(a) (3.0 points)

i. The structure of directories and files in FFS will always form a tree.

○ True

● False

ii. Explain.

FFS supports hard links, which allow a file to have multiple parents, so the structure forms a directed acyclic graph (DAG) but not a tree.

While not the intention of the question, a soft link or interpreting `.` and `..` as part of the structure would introduce cycles into the structure, not making it a tree regardless.

(b) (3.0 points)

i. A free space bitmap always uses less space than a free block list.

- ○ True
- ● False

ii. Explain.

When there is no free space, the free block list is empty and therefore takes no space.

(c) (3.0 points)

i. A simple file system, which stores files contiguously, suffers from external fragmentation.

- ● True
- ○ False

ii. Explain.

Since a simple file system stores files contiguously, it suffers from external fragmentation since if a block of memory isn't big enough, we can't use it to store this file.

(d) (3.0 points)

i. It is easy to make changes (e.g. adding IPv6 support) in the network layer.

- ○ True
- ● False

ii. Explain.

IP is the "narrow waist", the only network layer protocol that provides abstractions for the layers above (Applications, Transport), and the layers below (Data Link, Physical) Changing it would require changing layers above and below the IP layer as well.

(e) (3.0 points)

i. I/O bound tasks tend to have short CPU bursts.

- (●) True
- ( ) False

ii. Explain.

I/O bound tasks are interactive and so execute quickly.

(f) (3.0 points)

i. The end-to-end principle states that the design of a system should be guided by end-to-end examples.

- ○ True
- ● False

ii. Explain.

It is a suggestion on which layer/module should implement certain functionalities.

(g) (3.0 points)

i. Fate sharing means if two processes in a distributed system rely on each other, then they should be co-located onto the same machine.

◯ True

◯ False

ii. Explain.

Fate-sharing is about state placement.

This question was not graded due to this content not being covered during lecture.

2. (21.0 points) Select All

For the following questions, select all that apply. If none should be selected, don't select any. Selecting an incorrect choice or not selecting a correct choice will lead to point deductions, but the overall score for each question will be a minimum of 0.

(a) (3.0 pt) Which of the following are true about I/O and file systems?

- [x] Sequential reads from a hard drive are faster than random reads.
- [x] Memory mapped I/O is implemented by hardware mapping physical memory addresses to device registers.
- [ ] FFS organizes free blocks using a linked list.
- [ ] A file in NTFS must be stored contiguously on disk.
- [x] Executable files are read from disk using memory mapped I/O upon calling `exec`.

(b) (3.0 pt) Which of the following are true about networking?

- [ ] Congestion control ensures a fast sender does not overwhelm a slow receiver.
- [x] In TCP, the ACK packet contains the next sequence number that the receiver is expecting.
- [x] It is impossible to solve the Byzantine Generals problem with 1 malicious lieutenant among 3 total participants.
- [ ] HTTP is a protocol in the transport layer.
- [ ] TCP sessions survive changes of IP addresses.

(c) (3.0 pt) Which of the following are true about 2PC?

- [x] Once a worker has voted to abort, it is safe to proceed to abort its local update without waiting for the global decision.
- [x] Once the coordinator has received at least one abort vote, it is safe to immediately record in its log and send `GLOBAL-ABORT`.
- [ ] For one run of the protocol, the total messages sent between the coordinator and N workers (i.e. not counting the client) are exactly 4N (N vote requests, N votes, N global decisions, N ACKs).
- [ ] Logging is not a necessary mechanism for 2PC.
- [x] It is possible for 2PC to come to a decision even if multiple workers are dead.

(d) (3.0 pt) Which of the following are true about ACID properties?

- [x] Atomicity means each transaction should be treated as an indivisible unit (either all operations take effect, or none).
- [ ] Consistency means all workers/replicas in the system must have the same view of the system state.
- [x] Isolation means concurrent transactions do not interfere with each other.
- [ ] Durability guarantees the system is recovered to the correct state after a crash.
- [x] Isolation guarantees that transactions appear to be serialized.

(e) **(3.0 pt)** Which of the following are true about Pintos?

- ☐ All processes should start off by having the root directory as their current working directory.
- ■ Current working directories are process-specific, as opposed to being thread-specific.
- ☐ Reading and writing to different disk sectors can not be done concurrently.
- ■ Reading and writing to the same disk sector can not be done concurrently.
- ■ There cannot be two inodes with the same inode number within a file system.

(f) **(3.0 pt)** Which of the following are true about files?

- ☐ FAT file system can suffer from external fragmentation.
- ☐ FAT file system has good random access in comparison to extent or inode based file systems.
- ■ Empirical evidence suggests most files are small (less than 512K bytes).
- ☐ A file descriptor is equivalent to the inode number.
- ☐ An inode contains the file name in addition to direct, indirect, doubly indirect, and triply indirect pointers.

(g) **(3.0 pt)** Which of the following are true about reliability and networking?

- ■ With WAL, having idempotent operations makes it easy to maintain consistency.
- ■ With WAL, we can only apply changes to disk after they have been logged.
- ☐ When using WAL, we only replay operations that occurred before a COMMIT.
- ☐ UDP is a reliable, ordered, error-checked packet delivery system.
- ☐ In the internet layer module, each layer relies on the services from above.

3. (32.0 points) Short Answer

Answer each question in three sentences or less unless otherwise specified. You may not use semicolons to circumvent this limit. Any answer longer than three sentences will receive a 0. Any answer without explanation may not receive credit.

(a) (4.0 pt) In a file transfer program between two hosts over the internet, where would the conservative interpretation of the end-to-end principle dictate reliability be implemented? Explain.

The end-to-end principle dictates that verification of reliable transfer must be implemented at the two hosts (i.e. endpoints).

(b) (4.0 pt) When using FFS, explain what happens with regards to the disk as free space drops below 10-20%.

Fragmentation will occur as FFS will not be able to place contiguous blocks near each other ("failure to localize blocks" in paper). This will lead to poor disk throughput.

(c) (4.0 pt) Explain how data can be recovered from a single disk failure using RAID 5.

Parity blocks XOR data blocks across the other disk blocks within the same stripe unit. If one disk block fails, then that disk block can be reconstructed by XORing the rest of the disk blocks along with the parity block.

(d) (4.0 pt) Of the four necessary conditions for deadlock, which is the one most commonly eliminated? Explain.

Circular waiting is the most commonly eliminated by defining a global acquisition order as seen in Project 2.
Eliminating mutual exclusion necessitates creating more resources. Eliminating hold and wait or no preemption requires changing the core functionality of an operating system. All of these are less feasible compared to defining a global acquisition order.

(e) (4.0 pt) Explain one situation where busy waiting is appropriate.

Implementing locks on multiprocessors, very short critical sections, tightly interacting threads, or ultra-low latency I/O are all situations where busy waiting is appropriate.

(f) (4.0 pt) In 2PC, what condition(s) must be met for a transaction to be committed? What condition(s) must be met for a transaction to be aborted?

For a commit, the workers need to unanimously vote commit, and the coordinators need to log a `GLOBAL-COMMIT`. Other scenarios will lead to an abortion, such as the situation when the coordinator crashes before logging a `GLOBAL-COMMIT` and recovers.

(g) **(4.0 pt)** What is the major difference between write through and write back caches? Which approach are you supposed to implement in Project 3?

Write-through caching is when you update a block on disk at the same time you update the block in the cache. Write-back caching is when you only update a block on disk when you evict a block from the disk. In project 3, we used a dirty bit to implement write-back caching.

(h) **(4.0 pt)** There are two major protocols in the transport layer; one is best effort (packets can be lost, corrupted, and out of order), and another is reliable, ordered, and error-checked. Why would we ever want to use the best effort option?

The best effort option (UDP) is used in applications that prioritize speed and low overhead, and either don't care about being "lossy" or implements their own protocol for reliability. For example, in streaming audio or video, it doesn't matter if some packets are lost or corrupted, as long as most of the packets are sent, the user will still be able to hear/see the data well enough. Another example would be any application where real time data is very important (for example real time news, weather, stock price tracking, etc) and we don't have time for anything beyond best effort delivery. Another example would be DNS, which only involves a single UDP request and needs to be fast with low overhead.

4. (20.0 points) Word Hunters

Joy, Sophie, and Adina have recently been playing GamePigeon games. They need your help to ensure they play the same game using two-phase commit (2PC).

The trio will either choose to switch to a different game or continue playing the same game. If no action has been determined, they will continue playing the same game.

Joy is the coordinator while Sophie and Adina are workers. Currently, they are playing Word Hunt. Joy wants to switch to play Anagrams. Assuming no crashes and timeouts, here is the correct sequence of events.

```
1. Joy writes PREPARE to her log.
2. Joy sends Sophie VOTE-REQ to switch to Anagrams.
3. Sophie writes VOTE-COMMIT to her log.
4. Sophie sends Joy VOTE-COMMIT in favor of switching to Anagrams.
5. Joy sends Adina VOTE-REQ to switch to Anagrams.
6. Adina writes VOTE-COMMIT to her log.
7. Adina sends Joy VOTE-COMMIT in favor of switching to Anagrams.
8. Joy writes GLOBAL-COMMIT to her log.
9. Joy sends Sophie GLOBAL-COMMIT to switch to Anagrams.
10. Sophie writes GLOBAL-COMMIT to her log.
11. Sophie changes the game she is playing to Anagrams.
12. Sophie sends Joy ACK.
13. Joy sends Adina GLOBAL-COMMIT to switch to Anagrams.
14. Adina writes GLOBAL-COMMIT to her log.
15. Adina changes the game she is playing to Anagrams.
16. Adina sends Joy ACK.
17. Joy writes GOT-COMMIT to her log.
```

Only assume the situations specified in each question (i.e. situations don't carry over into the next question). Assume that all transmissions over the network are instantaneous and guaranteed to succeed.

Answer each question in three sentences or less unless otherwise specified. You may not use semicolons to circumvent this limit. Any answer longer than three sentences will receive a 0. Any answer without explanation may not receive credit.

(a) (4.0 points)

i. Sophie is very excited to switch games such that she sends Joy the `VOTE-COMMIT` before writing it in her log (i.e. switches steps 3 and 4). Could this be a problem?

- (●) Yes
- ( ) No

ii. Explain.

Sophie's decision must be recorded in the log before she sends her decision to Joy. If Sophie crashes after sending VOTE-COMMIT without logging it before, she won't remember what decision she made.

(b) (4.0 points)

i. If Joy crashes right after step 7, is the trio guaranteed to keep playing Word Hunt?

- (●) Yes
- ( ) No

ii. Explain.

When Joy reboots, she will see that she has not logged a global action but has started a vote request. As a result, she will send out a `GLOBAL-ABORT` to everyone, so everyone will still play Word Hunt.

(c) (4.0 points)

i. If Joy crashes right after step 1 and never recovers, are Sophie and Adina guaranteed to switch to Anagrams?

○ Yes

● No

ii. Explain.

Since Joy never transmitted the VOTE-REQ, Sophie and Adina will timeout and abort.

(d) (4.0 points)

i. Adina forgets to send Joy an ACK (i.e. step 16 does not happen). If Adina doesn't respond within the timeout, should Joy send a `GLOBAL-ABORT` to Sophie and Adina?

○ Yes

● No

ii. Explain.

Joy already decided to `GLOBAL-COMMIT`, so this transaction must take place. As a result, Joy will continue retrying `GLOBAL-COMMIT` until all `ACK`s are received.

(e) (4.0 points)

i. Joy wants to coordinate a specific time as well to play the game, so they decide to specify a time when sending messages and logging (e.g. "VOTE-REQ: Play Anagrams tomorrow"). Is this sufficient to handle the new time constraint?

- ● Yes
- ● No

ii. Explain.

With the guarantee of transmissions, the General's Paradox can be interpreted to be solved. The binary nature of the decision means that 2PC will still ensure system consistency.
On the other hand, the General's Paradox may still come into play depending on the recovery mechanisms of the participants. The possibility of participant crashes still exists since there was no such assumption about this. As a result, the system may not be Byzantine fault tolerant.

5. (20.0 points) Boosted Autograder

Kenneth wants to improve the average response time of the autograder. Suppose an autograder server processes each grade request in order as they arrive. All grade requests must wait in a single queue. Using a stopwatch, Kenneth determines that grade requests arrive in the autograder queue following a Poisson process. Moreover, he determines the service times are exponentially distributed. Therefore, the autograder queue is a M/M/1 queue.

He is exploring proposals to improve performance. Kenneth consults Michael and Nathan regarding some options he can choose.

Michael suggests adding another server. By adding another server, each individual autograder server will receive queries at a reduced $\frac{\lambda}{2}$ arrival rate with service time T_{service}.

On the other hand, Nathan suggests upgrading the current server. There will still be one server receiving requests at a rate of λ but it will complete each grade request twice as fast with service time $\frac{T_{\text{service}}}{2}$.

The following assumptions are made.

- The average interarrival time for grade requests is 10 minutes.
- The average service time is 8 minutes.
- Each server has its own queue with incoming requests to the autograder evenly balanced across each server's queue.

Help Kenneth figure out which design will best reduce the average response time.

Express your answer in a decimal with a leading zero if necessary (i.e. 0.162 instead of .162) and without any trailing zeros (i.e. 162 instead of 162.0). Round to two decimals if necessary.

Answer each question in three sentences or less unless otherwise specified. You may not use semicolons to circumvent this limit. Any answer longer than three sentences will receive a 0. Any answer without explanation may not receive credit.

(a) (4.0 points)

i. Kenneth wants to assess the current state of the autograder, so he calculates that the current autograder utilization is 0.8. Is this correct?

- [x] Yes
- [ ] No

ii. Explain.

$u = \lambda \times T_{\textbf{service}} = \frac{1}{10} \times 8 = 0.8$.

(b) (4.0 points)

i. Assume Kenneth correctly calculated the utilization of 0.8. What is the current average response time (in minutes) of the system (i.e. without any modifications)?

40

ii. Explain.

$T_{\textbf{sys}} = T_q + T_{\textbf{service}} = T_{\textbf{service}} \times \frac{u}{1-u} + T_{\textbf{service}} = 8 \times \frac{0.8}{0.2} + 8 = 40$**.**

(c) (4.0 points)

i. Assume Kenneth correctly calculated the utilization of 0.8. What is the current average length of the queue currently (i.e. without any modifications)?

3.2

ii. Explain.

$L_q = \lambda \times T_q = \frac{1}{10} \times 32 = 3.2$**.**

(d) (4.0 points)

i. Assume Kenneth correctly calculated the utilization of 0.8. What is the average response time (in minutes) if Kenneth takes Michael's suggestions?

13.33

ii. Explain.

By adding another server, the utilization will decrease by a factor of 2 as the arrival rate on any single queue will decrease by 2. Therefore, $T_{\text{sys}} = T_{\text{q}} + T_{\text{service}} = T_{\text{service}} \times \frac{0.5u}{1-0.5u} + 8 = 13.33$**.**

(e) (4.0 points)

i. Assume Kenneth correctly calculated the utilization of 0.8. What is the average response time (in minutes) if Kenneth takes Nathan's suggestions?

6.67

ii. Explain.

By upgrading the server, the service time will decrease by a factor of 2. With half the service time, utilization decreases to $u = 0.4$. Therefore, $T_{\text{sys}} = T_{\text{q}} + 0.5T_{\text{service}} = 0.5T_{\text{service}} \times \frac{0.5u}{1-0.5u} + 4 = 6.67$.

6. (34.0 points) Entertainment Systems

After his career as a 162 TA, Sean has been contacted by Korean entertainment companies to help build some distributed systems infrastructure to support their artists.

(a) (16.0 points) Instagram File System

EDAM Entertainment has asked Sean to design a file system for storing and receiving Instagram photos. Jieun wants to save her photos in high quality, which would result in 2048 photos, each of size 4 MiB. Each photo will be stored as an individual file.

Answer each question in three sentences or less unless otherwise specified. You may not use semicolons to circumvent this limit. Any answer longer than three sentences will receive a 0. Any answer without explanation may not receive credit.

i. (4.0 points)

A. Sean initially considers taking the simple route and implementing FAT32. Would he face any issues?

- ● Yes
- ● No

B. Explain.

Sean would not face any issues since each photo is 4 MiB which is well within the 4 GiB file size limit for FAT32.
Alternatively, performance can be an issue with FAT32. If the disk sector size is very small in comparison to the photo sizes, then random access and locality issues arise.

ii. (4.0 points)

A. From his experience with inodes in Pintos, Sean is also considering implementing a 32-bit inode-based file system with 4 KiB disk sector size and each inode is the size of one disk sector. Assuming there is sufficient disk space, is using only direct pointers sufficient to store a photo and maintain a functioning file system?

○ Yes

● No

B. Explain.

A 4 KiB disk sector size means there can be at most $2^{12}/2^2 = 2^{10} = 1024$ direct pointers in an inode. This would be exactly enough for a photo of 4 MiB since each direct pointer points to a 4 KiB disk sector, meaning 1024 of them would cover $2^{10} \times 2^{12} = 2^{22} = 4$ MiB. However, each inode has some metadata that needs to be stored as well, meaning we actually can't store 1024 direct pointers.

iii. (4.0 points)

A. In addition to the photos, Jieun also wants to store some high resolution videos, each of size 8 GiB. Sean is considering only using two triply indirect pointers and eliminating any other pointers. Will this introduce any problems?

- (●) Yes
- ( ) No

B. Explain.

While two triply indirect pointers is enough to support a video of size 8 GiB, reading in the smaller photos of size 8 MiB will be slower since triply indirect pointers require four disk accesses (triply indirect block, doubly indirect block, indirect block, and data block).

iv. (4.0 points)

A. Does Sean need to include any specific piece of metadata in an inode to implement hard links?

- (●) Yes
- ( ) No

B. Explain.

Sean needs to include a reference count in each inode to keep track of the number of hard links pointing at a particular inode.

(b) Nine Way Connection

In an effort to reunite all nine members of SNSD, SM Entertainment has asked Sean to setup the some networking infrastructure.

Answer each question in three sentences or less unless otherwise specified. You may not use semicolons to circumvent this limit. Any answer longer than three sentences will receive a 0. Any answer without explanation may not receive credit.

i. (4.0 points)

A. Sean first looks to setup a network for Taeyeon and Sooyoung who still spend a majority of their work day in the same office building in Korea. Which layer of the IP layering model will Sean be working in?

- ◯ Application
- ◯ Transport
- ◯ Network
- ● Datalink
- ◯ Physical

B. Explain.

Sean would be setting up a LAN since they are all in the same building (i.e. same geographical proximity).

ii. (4.0 points)

A. On the other hand, Yoona is currently in China filming her upcoming drama, so Sean is setting up a WAN for her as well to communicate with the team back in Korea. As Sean setups the routers, he realizes he only has information of the MAC address of Yoona's laptop. Will this pose a problem if Sean wants to build a scalable system?

- [x] Yes
- [ ] No

B. Explain.

The routers that make up the WAN forward packets using IP addresses not MAC addresses.

iii. (4.0 points)

A. With the newly setup networks, Taeyeon sends a snippet of her new song of 500 bytes to Yoona over the TCP setup by Sean. Sequence numbers for both Taeyeon and Yoona start at 0. Suppose the song is split up into 100 byte packets. What number should be in the ACK that Yoona sends to Taeyeon if Yoona has received bytes 1-200 and 301-400?

201

B. Explain.

The sequence number in the ACK represents the next byte that needs to be received. While bytes 301-400 have been received, the sequence number is the smallest byte that hasn't been received yet.

iv. (6.0 points)

A. Sean is also building a reliable file transfer system to help the members share their current songs and dance videos with the rest of the team. Due to the geographic spread of the members, each communication need to go through five links which each take 1 ms to cross. Sean has found that one of those links will lose the packet 50% of the times while the other ones succeed 100% of the times. Sean is willing to forego the end-to-end principle to reduce the expected time to successfully transmit a packet but only if he can cut it by at least half. Ignoring the costs of checks, should Sean forego the end-to-end principle and implement checks for each router in the network?

○ Yes

● No

B. Explain. You may use up to five sentences for this explanation.

Sean should stick with the end-to-end principle. The probability that a packet will successfully cross all five links is 50%. On expectation, it will take two tries to get across all packets, meaning $2 \times 5 = 10$ ms of expected latency when using the end-to-end principle. If each router checks on the other hand, it will take an expected two tries for just one of the routers, meaning $2 + 4 = 6$ ms of expected latency. This is not a 50% boost in performance, so Sean should adhere to the end-to-end principle.

7. (45.0 points) Directory Deep Dive

Being the GOAT he always is, Neil has finished Project 3 early and looking for something to do. He decides to poke around with some directory functionalities.

(a) (24.0 points) Deep Directories

Neil wants to implement `add_file`, which given an input string and a directory path, will recursively create nested directories for each character in the input string until the last letter, for which it will create a file. For example, if `input` is `"cs162rules!"` and `path` is the path of the current working directory, then `add_file` should create a file with a relative path of `"c/s/1/6/2/r/u/l/e/s/!"`.

```
/* Computes the length of the string S. */
size_t strlen(const char* s);

/* Copies the string SRC to DST (including the terminating null character).
   Returns DST. */
char* strcpy(char* dst, const char* src);

/* Writes characters into STR according to FORMAT. */
void sprintf(char* str, const char* format, ...);

/* Opens a new file PATH and access permission specified by MODE. Equivalent to
   open(path, O_CREAT | O_TRUNC | O_WRONLY, mode). Returns a file descriptor on
   success or -1 on failure. */
int creat(const char* path, mode_t mode);

/* Creates a new directory PATH with access permission specified by MODE.
   Returns 0 on success or -1 on error. */
int mkdir(const char* path, mode_t mode);

/* Contiguously allocates enough space for COUNT objects that are SIZE bytes of
   memory each and returns a pointer to the allocated memory. The allocated
   memory is filled with bytes of value zero. */
void* calloc(size_t count, size_t size);

/* Deallocates the memory allocation pointed to by PTR. */
void free(void* ptr);

/* Permissions for file and directory. */
mode_t mode_file = S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH;
mode_t mode_dir = S_IRWXU | S_IRWXG | S_IROTH | S_IXOTH;

void add_file(const char* path, const char* input) {
  if (_____[A]_____) {
    return;
  }

  char* new_path = calloc(_____[B]_____, sizeof(char));
  _____[C]_____;

  if (strlen(input) == 1) {
    _____[D]_____;
  } else {
    _____[E]_____;
    char* new_input = calloc(_____[F]_____, sizeof(char));
    _____[G]_____;
```

```
    _____[H]_____;
    free(new_input);
  }

  free(new_path);
}
```

Help Neil complete `add_file`! For the sake of simplicity, you do not need to worry about error checking. Assume all function calls succeed if given inputs are in the correct format, and that the input path represents a valid path to a directory.

You may only use methods and data structures given **in this specific code segment** (i.e. code presented immediately above). You may assume calls to any given library methods will succeed. Only **one piece of code** must be written per blank (i.e. no multiple lines with semicolons). Each blank **must contain code and cannot be a blank line**. When allocating stack variables, you must allocate the **minimum number of bytes** possible.

i. (3.0 pt) [A]

```
strlen(input) == 0
```

ii. (3.0 pt) [B]

```
strlen(path) + 3
```

iii. (3.0 pt) [C]

```
sprintf(new_path, "%s/%c", path, input[0])
```

iv. (3.0 pt) [D]

```
creat(new_path, mode_file)
```

v. (3.0 pt) [E]

```
mkdir(new_path, mode_dir)
```

vi. (3.0 pt) [F]

```
strlen(input)
```

vii. (3.0 pt) [G]

```
strcpy(new_input, &input[1])
```

viii. (3.0 pt) [H]

```
add_file(new_path, new_input)
```

(b) (21.0 points) Recursive Removal

After running his first method too many times, Neil finds himself with very deep directories. He tries to delete them from the root directory but runs into an issue from Project 3 where he was not allowed to remove non-empty directories. He decides to implement `rmdir_recursive` for recursive directory removal (i.e. similar to `rm -r`) and needs your help yet again.

This means that we delete every entry within the directory, and if the entry is a subdirectory, in addition to deleting the subdirectory itself, we also delete every entry within that subdirectory, and every entry in those entries, etc. Assume each directory by default has two entries `.` and `..` in it, which point to the current directory and its parent directory, respectively.

```
typedef struct dirent {
  ...
  char name[FILENAME_MAX];
  bool is_dir;
  ...
} dirent;

/* Opens directory named by FILENAME and returns a pointer to the stream. */
dir* opendir(const char* dirname);

/* Closes the file descriptor associated with DIRP. */
int closedir(dir* dirp);

/* Returns a pointer to the next directory entry or NULL upon reaching the end
   of the directory or on an error. */
dirent* readdir(dir* dirp);

/* Removes a directory file whose name is given by PATH. The directory must not
   have any entries other than "." and "..". */
int rmdir(const char* path);

/* Removes file or directory specified by PATH. */
int remove(const char* path);

/* Computes the length of the string S. */
size_t strlen(const char* s);

/* Lexicographically compares S1 and S2. Returns an integer greater than, equal
   to, or less than 0, according as the string S1 is greater than, equal to, or,
   less than the string S2. */
int strcmp(const char* s1, const char* s2);

/* Writes characters into STR according to FORMAT. */
void sprintf(char* str, const char* format, ...);

void rmdir_recursive(const char* path) {
  dir* directory = opendir(path);
  struct dirent* entry;

  while ((_____[A]_____)) {
    if (_____[B]_____) {
      continue;
    }

    char entry_path[_____[C]_____];
```

```
    _____[D]_____;

    if (entry->is_dir) {
      _____[E]_____;
    } else {
      _____[F]_____;
    }
  }

  closedir(directory);
  _____[G]_____;
}
```

Help Neil complete `rmdir_recursive`! For the sake of simplicity, you do not need to worry about error checking. Assume all function calls succeed if given inputs are in the correct format, and that the input path represents a valid path to a directory.

You may only use methods and data structures given **in this specific code segment** (i.e. code presented immediately above). You may assume calls to any given library methods will succeed. Only **one piece of code** must be written per blank (i.e. no multiple lines with semicolons). Each blank **must contain code and cannot be a blank line**. When allocating stack variables, you must allocate the **minimum number of bytes** possible.

i. (3.0 pt) [A]

```
entry = readdir(directory)
```

ii. (3.0 pt) [B]

```
strcmp(".", entry->name) == 0 || strcmp("..", entry->name) == 0
```

iii. (3.0 pt) [C]

```
strlen(path) + strlen(entry->name) + 2
```

iv. (3.0 pt) [D]

```
sprintf(entry_path, "%s/%s", path, entry->name)
```

v. (3.0 pt) [E]

```
rmdir_recursive(entry_path)
```

vi. (3.0 pt) [F]

```
remove(entry_path)
```

vii. (3.0 pt) [G]

```
rmdir(path)
```

`remove(path)` is also acceptable since `path` is a directory, so this is equivalent to `rmdir(path)`.

8. References

(a) Unit Conversions

Prefix	Factor
Gibi (Gi)	2^{30}
Mebi (Mi)	2^{20}
Kibi (Ki)	2^{10}
milli (m)	10^{-3}
micro (μ)	10^{-6}
nano (n)	10^{-9}

(b) Policy Acronyms

Acronym	Term
FAT	File Allocation Table
FFS	Fast File System
NTFS	New Technology File System
WAL	Write Ahead Logging
2PC	Two Phase Commit

9. Clarifications

True or False

- (1.g) Ignore this question. It will not be graded.

Select All

- (2.a) Assume BSD 4.2 Berkeley FFS for the third item.
- (2e) Assume Pintos after correctly implementing Project 3.

Short Answer

- (3.b) Assume BSD 4.2 Berkeley FFS.

Word Hunters

- (4.a) When asking could this be a problem, not being able to play Anagrams is not a problem.
- (4.e) Assume an exact absolute time is given with timezone.

Boosted Autograder

- (5.b-5.e) Assume utilization calculated is for the situation in (5.a).

Entertainment Systems

- (6.a.ii) Assume 4-byte pointers.
- (6.1.iii) Assume the same situation as 6.1.2.
- (6.b.iii) The bytes of Taeyeon's message are indexed starting at 1.
- (6.b.iv) Assume the link that is 50% successful is fixed (i.e. not random) and known to Sean.
- (6.b.iv) You may assume the network can only handle one packet at a time.

Directory Deep Dive

- Heap variables must use min number of bytes.
- (7.b) You may use `strlen` from (7.1)']

No more questions.