Agenda for Today

1. Remote Work Update
2. Assignment and Lab Q&A
3. Introduction to POSIX
4. Processes
5. Wrap Up
Remote Work Update
Q&A
What happens if we have an input string that contains newlines?
   A: The ngram will contain a newline character, so it will print over multiple lines. That’s fine—you don’t have to do anything special to print it.

Do we need a header with our name, date, class, academic honesty statement, etc.?  
   A: No, not for assignments.

Do we have to read input in multi-character buffers with fgets, or can we just read one character at a time?  
   A: You can do either.
Warmup Lab Q&A

Do both group members have to turn in the lab, or just one person?
A: Only one group member needs to turn in the lab.

PLEASE ADD YOUR GROUP ON GRADESCOPE.

Will a line of text ever be longer than the buffer we use for fgets?
A: Yes, almost certainly. No fixed size buffer will be large enough for all possible inputs. You should plan on filling your buffer multiple times.

Upcoming Office Hours:
Thursday: Isa at 11am, Charlie at 3pm
Friday: Charlie at 11am
Answer these questions about your function:

What are the function's parameters?

*Explain what they mean, not just their types.*

What does the function do?

*Think about when and why you might use this function.*

*Explain it like you're talking to a 161 student.*

What does the function return?

*Both success and error return values matter.*
```c
int open(const char* path, int oflag, ...);
```

**What are the function's parameters?**
- `path` is a string that says where to find the file we want to open.
- `flags` is an integer with *many* options like `O_CREAT`, which add special functionality.
  If you specify `O_CREAT`, a third parameter (mode) says what permissions a created file should have.

**What does the function do?**
- The open function opens a file. An opened file can be read, written, etc.
- Some extra functionality:
  - `O_CREAT` may create a file
  - `O_TRUNC` will clear the opened file's content

**What does the function return?**
- `open` returns a file descriptor (a non-negative integer) on success.
- If `open` fails, it returns -1 (and sets `errno`, whatever that is).
FILE* fopen(const char* path, const char* mode);

What are the function's parameters?

path: where is the file we want to open? mode: “r”, “r+”, “w”, “w+”, “a”, or “a+”
Mode controls how the file is opened.

What does the function do?

Pretty much the same as open(), but with different options.
The “+” mode options create the file as needed.
The “a” and “a+” options open the file in append mode.

What does the function return?

On success: returns a FILE*, otherwise NULL.
A FILE* uses buffered reading and writing; not all reads or writes actually get new data.
```c
ssize_t read(int filedes, void* buf, size_t nbytes);
```

**What are the function's parameters?**

- A file descriptor (filedes)
- `buf`: a buffer we will read up to `nbytes` into
- `nbytes`: how many bytes will be read?

**What does the function do?**

Attempts to read up to `nbytes` bytes into the buffer.
The file has an offset, which is the current position in the file.
After characters are read, the offset is moved forward in the file.

**What does the function return?**

`read` returns the number of bytes read (may be smaller than `nbyte`)
If `read` returns zero, the file offset was at the end of the file.
On error, `read` returns -1 and sets `errno`. 
void perror(const char *s);

What are the function's parameters?
s is a string, which perror prints right away.

What does the function do?
perror prints the string it was given if it is not null
Then, perror prints a string corresponding to the most recent error set in errno

What does the function return?
No return value. The only output is the error message that is printed.
```c
int snprintf(char* str, size_t size, const char* format, ...);
```

**What are the function's parameters?**

- A character array `str` with space for `size` bytes
- A string that uses printf-style formatting
- As many extra values as you want to print, specified by the format string.

**What does the function do?**

- `snprintf` stores the first size-1 characters of formatted string output to `str` (the output buffer)
- If the formatted output is too long, it is cut off at `size` characters.

**What does the function return?**

- The return value is the number of characters successfully printed, or if the output is truncated, `snprintf` returns the number of characters that could be printed if there were more space. The `str` output buffer includes a null terminator after the output.
`ssize_t getline(char** linep, size_t* linecapp, FILE* stream);`

What are the function's parameters?
A pointer to a string buffer, a pointer to the size of the buffer, and a FILE stream

What does the function do?
getline reads an entire line of text from a FILE stream.
If the buffer `*linep` is too small or NULL, getline will grow the buffer and update the size at `*linecapp`

What does the function return?
The return value is the number of characters read, but not including the \0 terminator
return -1 if the read fails.
`*linep` now points to the string that was read.
char* strpbrk(const char* s, const char* charset);

What are the function's parameters?
The first string: we are searching through this string
The second string: a set of characters we are searching for

What does the function do?
strpbrk searches through the first string for the first occurrence of any of the characters in the second string.

What does the function return?
strpbrk returns a pointer to the first character it found from the `charset` input. If there are no matching characters, the return value is NULL.
void* memmove(void *dst, const void *src, size_t len);

What are the function's parameters?

two pointers to any type: a destination and a source
an integer: how many bytes should be moved

What does the function do?

memmove copies `len` bytes from `src` to `dst`. If the memory areas overlap, memmove will copy the values to a temporary location before writing to the destination.
memcpy will refuse to copy overlapping regions

What does the function return?

memmove returns a pointer to the memory location that was copied to (the destination)
memmove can’t fail (except maybe by crashing your program)
Processes
Limited Direct Execution

What does the book mean by "limited direct execution?"

Direct Execution: programs run directly on the CPU/hardware
Limited: protect system by limiting the time and operations that a program can do on its own

Why is this model useful?

You have to allow programs to run efficiently, but the OS still maintains security by constraining programs. This keeps programs from misbehaving.

How does it work?

There are multiple possible approaches, but one uses system calls:
Most of the time, programs run without the OS’s involvement. Periodically, programs need to perform operations that require special privileges (like opening files, writing or reading data, starting more programs, etc.) and program makes a system call to request that the OS do this on its behalf.
Library Calls vs. System Calls

Userspace

Program

libc

call `printf`

call `open`

call `read`

OS Kernel

Linux
Process Basics

What is a process?

Why do we have processes?

How do you create a process?
fork, wait, and exec
Wrap Up
Upcoming Work

Read for Class
   Review the shell lab before Friday's class

Work Due
   The ngram assignment and warmup lab are due Friday evening