Reading input in C

Adapted from materials by Dr. Carrier

Rishi - Unsplash

Where do we get input?

Where do we get input?

- stdin
- Command line arguments
- Files

Where do we get input?

- **stdin** (our focus here)
- Command line arguments
- Files

scanf

scanf

```
scanf(format_str, mem_addr_1, mem_addr_2, …);
```

scanf

```
scanf(format_str, mem_addr_1, mem_addr_2, …);
```

- format_str is a formatted string like in printf
  - E.g., “%d %f” for an int, space, float

scanf

```
scanf(format_str, mem_addr_1, mem_addr_2, …);
```

- format_str is a formatted string like in printf
  - E.g., “%d %f” for an int, space, float
- Each specified var (e.g., %d), will need a memory address

scanf

```
scanf(format_str, mem_addr_1, mem_addr_2, …);
```

- format_str is a formatted string like in printf
  - E.g., “%d %f” for an int, space, float
- Each specified var (e.g., %d), will need a memory address

- scanf returns a EOF (constant) if EOF (Ctrl+D) is sent via stdin
  - We can use this to know when the user wants to stop

fgets

```
fgets(char* s, int size,FILE* stream);
```

fgets

```
fgets(char* s, int size,FILE* stream);
```

- For now, our stream is stdin (built in)

fgets

```
fgets(char* s, int size,FILE* stream);
```

- For now, our stream is stdin (built in)
- We need to allocate char buffer

fgets

```
fgets(char* s, int size,FILE* stream);
```

- For now, our stream is stdin (built in)
- We need to allocate char buffer
- Will read, at most, n-1 chars
  - Null terminator (\0) placed after last char read

fgets

```
fgets(char* s, int size,FILE* stream);
```

- For now, our stream is stdin (built in)
- We need to allocate char buffer
- Will read, at most, n-1 chars
  - Null terminator (\0) placed after last char read
- Stops at newline or EOF

fgets

```
fgets(char* s, int size,FILE* stream);
```

- For now, our stream is stdin (built in)
- We need to allocate char buffer
- Will read, at most, n-1 chars
  - Null terminator (\0) placed after last char read
- Stops at newline or EOF
- Returns s if successful, NULL if not

getline

```
getline(char** lineptr, size_t *n,FILE* stream);
```

getline

```
getline(char** lineptr, size_t *n,FILE* stream);
```

- For now, our stream is stdin (built in)

getline

```
getline(char** lineptr, size_t *n,FILE* stream);
```

- For now, our stream is stdin (built in)
- Not in C standard, part of POSIX > 2008

getline

```
getline(char** lineptr, size_t *n,FILE* stream);
```

- For now, our stream is stdin (built in)
- Not in C standard, part of POSIX > 2008
- Delimits by newline

getline

```
getline(char** lineptr, size_t *n,FILE* stream);
```

- For now, our stream is stdin (built in)
- Not in C standard, part of POSIX > 2008
- Delimits by newline
- Reallocates memory if more room is needed

getline

```
getline(char** lineptr, size_t *n,FILE* stream);
```

- For now, our stream is stdin (built in)
- Not in C standard, part of POSIX > 2008
- Delimits by newline
- Reallocates memory if more room is needed
- Allocates memory for you if lineptr is NULL and n=0
  - Updates both values

getline

```
getline(char** lineptr, size_t *n,FILE* stream);
```

- For now, our stream is stdin (built in)
- Not in C standard, part of POSIX > 2008
- Delimits by newline
- Reallocates memory if more room is needed
- Allocates memory for you if lineptr is NULL and n=0
  - Updates both values
- Returns the number of chars read
  - Or -1 for errors