

Discussion Session

01.22.2025

Project 1 - Example

sing o goddess the anger of achilles son of peleus that brought countless ills upon the achaeans
many a brave soul did it send hurrying down to hades
many a hero did it yield a prey to dogs and vultures
for so were the counsels of jove fulfilled from the day on which the son of atreus king of men
and great achilles first fell out with one another
and which of the gods was it that set them on to quarrel

```
$ ./sortk input.txt 1
```

and great achilles first fell out with one another
and which of the gods was it that set them on to quarrel
for so were the counsels of jove fulfilled from the day on which the son of atreus king of men
many a brave soul did it send hurrying down to Hades
many a hero did it yield a prey to dogs and vultures
sing o goddess the anger of achilles son of peleus that brought countless ills upon the achaeans

Project 1

- You will make a program named “sortk”.
- It will sort inputted sentences with k-th word in the sentence.
- Familiarize yourself with C programming!
- Get used to linux environment!

Project 1 - Basic Workflow

1. Parse command line arguments
 - a. Look for the meaning of argc, argv
2. Open the input file.
 - a. You may use fopen() or open()
3. Read lines from the input file
 - a. You may use fgets() getline() or read()
 - b. You should dynamically allocate memory to save lines.
4. Sort lines based on k-th word.
 - a. You may use qsort() to sort
 - b. You may use strtok() to find k-th word.
5. Print output.

man strtok

The strtok() function breaks a string into a sequence of zero or more nonempty tokens. On the first call to strtok(), the string to be parsed should be specified in str. In each subsequent call that should parse the same string, str must be NULL.

...

Each call to strtok() returns a pointer to a null-terminated string containing the next token. This string does not include the delimiting byte. If no more tokens are found, strtok() returns NULL.

...

and strtok() saves a pointer to the following byte;

...

man getline

`getline()` reads an entire line from stream, storing the address of the buffer containing the text into `*lineptr`. The buffer is null-terminated and includes the newline character, if one was found.

If `*lineptr` is set to `NULL` and `*n` is set 0 before the call, then `getline()` will allocate a buffer for storing the line. This buffer should be freed by the user program even if `getline()` failed.

Get used to linux environment!

- All the projects will be graded on CSL machines.
- Highly recommended to work your projects on Linux machine.
- **Do not work on WINDOWS machine!!! (WSL is okay)**
 - Windows uses CRLF at the end of line.
However, Linux uses LF. Working on Windows will mess up all the files.
 - WSL is fine, but please test on CSL machine before submitting.
- Please test on CSL machine before submitting
 - Even if you are using linux, program might behave differently with different compilers and environments.

Basic UNIX commands - Directories / Files

- ls - list directory contents
- cd - switch current directory
- pwd - display the present working directory
- cat [files] - concatenate and display [files]
- mkdir / rmdir - make / remove directory
- rm [files] - remove [files]
- mv [A] [B] - move [A] to [B]
- cp [A] [B] - copy [A] to [B]

Basic UNIX commands - Permission

- Representation : $\text{d} \underbrace{\text{rwx}}_{\text{u}} \underbrace{\text{r-x}}_{\text{g}} \underbrace{\text{r-x}}_{\text{o}}$

first character: file type ('-' regular 'd' directory 'l' symlink)

u: user who owns this file

g: users in the same group as the owner

o: all other users

r: readable w: writable x: executable

- Octal representation : 755, 777...

Basic UNIX commands - Permission

- Changing permission - chmod
- `chmod [mode] [file]`
- mode :
 - User-category: u,g,o, a(all)
 - +: add -: remove =: explicitly set the mode
 - rwx : permissions(read/write/execute)
 - Also supports octal mode.
- ex:
 - `chmod a+x file`
 - `chmod 640 file`

Basic UNIX commands - IO redirection and Pipe

- Shell can redirect stdin/stdout/stderr of a program to/from a different location
- Redirections : '>' and '<'
 - `prog > fileX` : redirect stdout of prog to fileX
 - `prog < fileX` : feed fileX to stdin of prog
 - `prog >> fileX` : append stdout at the end of fileX
 - `prog 2> fileX` : redirect stderr (fd : 2) of a prog to fileX
- Pipe: '|'
 - Pass the output from the previous command to the input of the next command
 - ex: `cat file.txt | grep CS537`

Quick recap to C programming

- Support structured programming
- Support development of the Unix OS and tools
 - As Unix became popular, so did C
- Implications for C
 - Good for system-level programming
 - But also used for application-level programming
 - Low-level
 - Close to assembly language; close to machine language; close to hardware
 - Efficiency over portability

Dynamic memory management

- A program often needs to allocate memory dynamically
- Use dynamic memory allocation
 - malloc - allocated memory
 - calloc - allocate cleared memory
 - realloc = resize previously allocated block of memory
 - returns NULL pointer if fails to allocate
- Programmers should free after use to avoid memory leak
 - Some languages have garbage collector, but C doesn't
 - Also should not double-free or use-after-free.

struct

- A collection of related data items.
Elements can have different types.
- To access elements:
use “.student_id” or “->student_id”.
- Typically located next to each element,
but memory layout might vary (gaps due to alignment...)
- Also used implementing DS like linked list
 - Search “linux kernel linked list” to see how it is used

```
struct student {  
    int student_id;  
    char name[NAME_LEN];  
};
```

Debugging with GDB

- Build with -g flag (`gcc -g -Og main.c`) (-Og to avoid optimization)
- Run with gdb to debug
- Commonly used commands
 - `run [args]` - run executable
 - `break [location]` - set breakpoint to [location]. You can add location as function name, line number...
 - `continue, step` - when paused for breakpoint, continue/step execution
 - `print [variable]` - see the variable.
 - `where` - see call stack, “up” / ”down” to move.
 - and so on... Try searching “gdb cheatsheet”