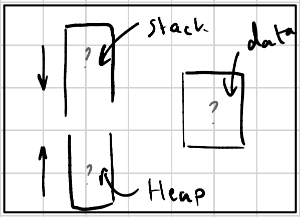
CS202 - Review Session 03

Agenda

1. C Review ✓
   - 1.1 Stack and Heap Memory Allocation ✓
   - 1.2 String Concatenation ✓
   - 1.3 Struct ✓
2. Lab 2 Overview ✓
   - 2.1 Motivation ✓
   - 2.2 File Definition ✓
   - 2.3 File Permissions ✓
   - 2.4 Flags ✓
   - 2.5 Functions ✓
   - 2.6 Helper Functions ✓
   - 2.7 Test Output ✓
3. Q & A

1. C Review

1.1 Stack and Heap Memory Allocation



* memory

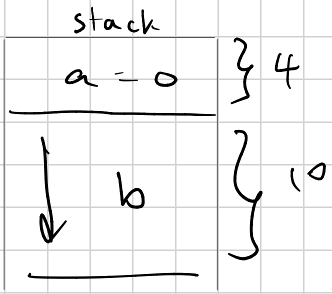
1. stack
2. heap
3. data

For example...

```
int main () {
    // Stack allocation
    int a = 0;
    char b[10];

    // Heap allocation
    int *p = malloc (sizeof (int));
```

malloc return type: void * or 0

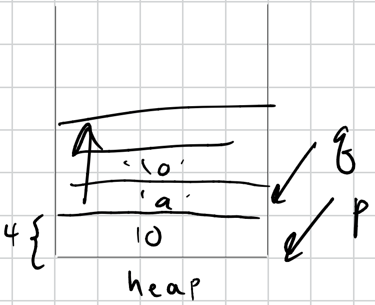


```
    if (p == 0) {
        exit (1);
    }
    *p = 10;

    char *q = malloc (5 * sizeof (char));
    if (q == 0) {
        exit (1);
    }
    q[0] = 'a';
    q[1] = '\0';
    printf ("%s\n", q);

    free (p);
    free (q);

    return 0;
}
```



1.2 String Concatenation

`int snprintf (char *str, size_t size, const char* format ...)`

- **char *str**: buffer, output after concatenation
- **size_t size**: max size to write into buffer
- **char *format**: format for concatenation, similar to how you would use printf.
- **...**: optional args (variables to be formatted like in printf)

For example...

```
const char *name = "Abigail"
printf ("Hello, my name is %s.\n", name);

// bad example
char* buffer;
```

```
snprintf(buffer, 200, "Hello, my name is %s.\n", name);

// correct example
char buffer[200];
snprintf(buffer, 200, "Hello, my name is %s.\n", name);
```

1.4 Struct

```
struct student {
    int age;
    char* name;
}


struct student alice
alice.age = 22;
alice.name = "Alice";

struct student *palice = &alice;
```

Access members of struct?

```
    alice.age;     OR     palice -> age;
```

2. Lab 2 Overview

2.1 Motivation

ls

2.2 File Definition

Files: "test.txt", "main.c"

Directories: "foo/", "bar/", "foo/bar/", ...

".git" ?

```
                                          ../../     ../
ls -a        b/hi/my/relative/path/to/file
             ↳b/no        cd ../no
"."           "./my/relative/path/to/file"
".."          "~/my/absolute/path"
           /                  ↑
```

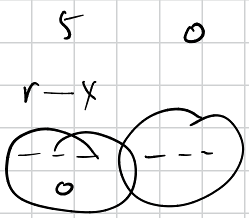
2.3 File Permissions

rwx --- ---

For example ...

```
chmod 700 file.txt
```

$700_8 \to 7, 0, 0 \to \underline{1}\,\underline{1}\,\underline{1}, \underline{0}\,\underline{0}\,\underline{0}, \underline{0}\,\underline{0}\,\underline{0}_2 \to$ rwx



Ex.
I want owner to read, write.
I want group and others to read only.

rw-, r--, r-- $\longrightarrow$ $110, 100, 100_2 \to 644$

Answer: chmod 644 file.txt

* Regarding directories...
- r - ls, list files
- w - Create / delete files
- x - cd !

2.4 Flags

$\underbrace{\texttt{./ls}}_{\text{program}}$ $\underbrace{\texttt{-alR}}_{\text{flags}}$ $\underbrace{\texttt{foo/ bar/}}_{\text{args}}$ ...

-alR
- → -a
- → -l
- → -R

man 3 getopt_long

int getopt_long(int argc, char* const argv[], const char* optstring,
                const struct option *longopts, int *longindex)

- argc: # of args supplied (from main)
- argv: array of args supplied (from main)
- optstring: the flag we want to parse
- longopts: ptr to option struct
- longindex: if not NULL, will set relative to longopts arr

Possible return values:

- option character → updates optind → on the next call, getopt() can resume the scan.
- -1 (otherwise)

man 3 getoptlong

"--" v.s. "-"

Functions

opendir

readdir

closedir

2.5 Helper Functions

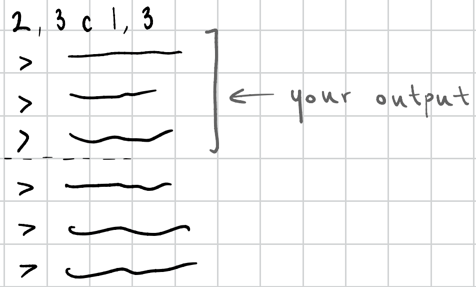
PRINT_PERM_CHAR
uname_for_uid
group_for_uid
date_string
ftype_to_string

*Please read through them carefully!

2.7 Test Output

diff your_output.txt system_output.txt

For example...



c :	change
a :	add
d :	delete