C for Operating Systems

printf

```c
void printfEx() {
  char c = 'a';
  char *str = "Kris Kool";
  int i = 10;
  double pi = 3.1415926;
  printf("character: %c, string: %s, int: %d, double: %f \n", c, str, i, pi);
  // cannot modify str, segmentation fault
  // str[3] = '\0';
  char *dest = (char *) malloc(80 * sizeof(char));
  strcpy(dest, str);
  dest[3] = '\0';
  printf("shorter string: %s\n", dest);
  free(dest);
}
```

Output:

```
character: a, string: Kris Kool, int: 10, double: 3.141593
shorter string: Kri
```

fopen

```c
void fopenEx() {
  FILE *fp = fopen("main.c", "r");
  assert(fp != NULL);
  char *buffer = NULL;
  size_t buffer_size = 0; // unsigned long
  ssize_t read;   // long
  while ((read = getline(&buffer, &buffer_size, fp)) != EOF) {
    // z - for size_t, u - for unsigned
    printf("Retrieved line of length %zu: \n", read);
    printf("%s", buffer);
  }
  free(buffer);
  fclose(fp);
}
```

Output:

```
Retrieved line of length 20:
#include <assert.h>
Retrieved line of length 19:
#include <stdio.h>
Retrieved line of length 20:
#include <stdlib.h>
...
```

command line arguments

```c
int main(int argc, char *argv[]) {
  for (int i = 0; i < argc; ++i) {
    printf("argument %d: %s\n", i, argv[i]);
  }
  fprintf(stderr, "printing to standard output\n");
  exit(EXIT_SUCCESS);
}
```

Output:

```
$ gcc main.c
$ ./a.out hello world this is 1 2 3
argument 0: ./a.out
argument 1: hello
argument 2: world
argument 3: this
argument 4: is
argument 5: 1
argument 6: 2
argument 7: 3
$
```

...

redirecting output

```c
int main(int argc, char *argv[]) {
  for (int i = 0; i < argc; ++i) {
    printf("argument %d: %s\n", i, argv[i]);
  }
  fprintf(stderr, "printing to standard error\n");
  exit(EXIT_SUCCESS);
}
```

Output:

```
$ gcc main.c
$ ./a.out > output.txt 2> errors.txt
$ cat output.txt
argument 0: ./a.out
$ cat errors.txt
printing to standard error
$ ./a.out > alloutput.txt 2>&1
$ cat alloutput.txt
printing to standard error
argument 0: ./a.out
$
```

getline - 1

```c
void getlineEx() {
  char *temp; char *original;
  char *line = NULL;
  size_t len = 0; // unsigned long
  ssize_t read;   // long
  while ((read = getline(&line, &len, stdin)) != EOF) {
    printf("line: %s", line);
    // strdup makes a copy
    temp = original = strdup(line);
    // temp is modified
    char *name = strsep(&temp, ";");
    int length = atoi(strsep(&temp, ";"));
    int height = atoi(strsep(&temp, ";"));
    printf("Name: %s, length: %d, height: %d \n", name, length, height);
    assert(temp == NULL);
    // duplicated from strdup
    free(original);
  }
  // resized by getline
  free(line);
}
```

getline - 2

```
...
    temp = original = strdup(line);
    // temp is modified
    char *name = strsep(&temp, ",");
    int length = atoi(strsep(&temp, ","));
    int height = atoi(strsep(&temp, ","));
...
}
```

Output:

```
$ ./a.out
square;2;5
line: square;2;5
Name: square, length: 2, height: 5
rectangle;   20      ; 30
line: rectangle;   20      ; 30
Name: rectangle, length: 20, height: 30
pentagon       ;    3   ; 7
line: pentagon       ;    3   ; 7
Name: pentagon       , length: 3, height: 7
```

"square;2;5" is typed by the user

extra space in name

tokenize

```
void tokenizeEx() {
  char line[80] = "A sentence with spaces";
  char *tokens[20];
  char *pch;

  pch = strtok(line, " ");
  int num = 0;
  while (pch != NULL) {
    printf("Token %d: %s\n", num, pch);
    tokens[num] = pch;
    ++num;
    pch = strtok(NULL, " ");
  }
}
```

Output:

```
$ ./a.out
Token 0: A
Token 1: sentence
Token 2: with
Token 3: spaces
```

memset / memcpy

```c
void memsetEx() {
  char str[20] = "Hello World";
  // Fill 3 characters starting from str[5] with '.'
  memset(str + 5, '.', 3 * sizeof(char));
  printf("Modified: %s\n", str);
  // change all to x
  memset(str, 'x', strlen(str) * sizeof(char));
  printf("x'ed: %s\n", str);
  char *world = "World";
  char *hello = "Hello";
  memcpy(str, world, strlen(world));
  memcpy(str + 6, hello, strlen(hello));
  printf("Recreated: %s\n", str);
}
```

Output:

```
$ ./a.out
Modified: Hello...rld
x'ed: xxxxxxxxxxx
Recreated: Worldxxxxxx
```

string sort - 1

```
// must use strcmp to compare char*
// common mistake is to use
// a < b or a == b
// which compares address of a and address of b
bool comesBefore(char *a, char *b) { return strcmp(a, b) < 0; }

void sortCharArray(char *arr[], int n) {
  for (int i = 0; i < n; ++i) {
    for (int j = i + 1; j < n; ++j) {
      if (comesBefore(arr[j], arr[i])) {
        char *temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
      }
    }
  }
}
```

string sort - 2

```
void strSortEx() {
  printf("Enter 5 strings\n");
  size_t buffer_size = 80;
  char *buffer = (char *)malloc(buffer_size * sizeof(char));
  ssize_t read; // long
  char *strArray[5];
  for (int i = 0; i < 5; ++i) {
    read = getline(&buffer, &buffer_size, stdin);
    buffer[read - 1] = '\0'; // get rid of \n
    strArray[i] = (char *)malloc(read * sizeof(char));
    strcpy(strArray[i], buffer);
  }
  free(buffer);
  sortCharArray(strArray, 5);
  for (int i = 0; i < 5; ++i) {
    printf("%s ", strArray[i]);
    free(strArray[i]);
  }
  printf("\n");
}
```

```
Output:
$ ./a.out
Enter 5 strings
hello
world
apple
orange
me
apple hello me orange world
```

memory leaks

- Each malloc requires free
- getline will resize buffer given, requires free
- strdup duplicates given char*, requires free
  - need a pointer to original of using strsep

```
void memleakEx() {
  char *str = (char *)malloc(100 * sizeof(char));
  char *buffer = NULL;
  size t buffer size;
  getline(&buffer, &buffer size, stdin);
  printf("String: %s\n", buffer);
}
```

Console:

```
$ gcc -fsanitize=address -fno-omit-frame-pointer main.c
$ ./a.out
hello world
==16145==ERROR: LeakSanitizer: detected memory leaks
Direct leak of 120 byte(s) in 1 object(s) allocated from:
    #0 0x7f2eebfb2f08 in __interceptor_malloc /tmp/gcc-9.2.0/libsanitizer/asan/asan_malloc_linux.cc:144
    #1 0x7f2eebb4d8d4 in _IO_getdelim (/lib64/libc.so.6+0x6f8d4)
    #2 0x402406 in memleakEx (/home/NETID/pisan/courses/430/c-for-os/a.out+0x402406)
    #3 0x40249d in main (/home/NETID/pisan/courses/430/c-for-os/a.out+0x40249d)
    #4 0x7f2eebb00504 in __libc_start_main (/lib64/libc.so.6+0x22504)

SUMMARY: AddressSanitizer: 220 byte(s) leaked in 2 allocation(s).
```

valgrind

- Each malloc requires free
- getline will resize buffer given, requires free
- strdup duplicates given char*, requires free
  - need a pointer to original of using strsep

Console:

```
$ gcc -g main.c
$ valgrind ./a.out
String:

==16303==
==16303== HEAP SUMMARY:
==16303==     in use at exit: 220 bytes in 2 blocks
==16303==   total heap usage: 2 allocs, 0 frees, 220 bytes allocated
==16303==
==16303== LEAK SUMMARY:
==16303==    definitely lost: 220 bytes in 2 blocks
==16303==    indirectly lost: 0 bytes in 0 blocks
==16303==      possibly lost: 0 bytes in 0 blocks
==16303==    still reachable: 0 bytes in 0 blocks
==16303==         suppressed: 0 bytes in 0 blocks
==16303== Rerun with --leak-check=full to see details of leaked memory
==16303==
==16303== For counts of detected and suppressed errors, rerun with: -v
==16303== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

static analysis

- clang++ for C++ files
- clang not as good for C files

Console:

```
$ clang++ --analyze main.cpp

main.cpp:3:19: warning: Value stored to 'c' during its initialization is
never read
void ex() { char *c = (char *) malloc(100 * sizeof(char)); }
                  ^   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
main.cpp:3:60: warning: Potential leak of memory pointed to by 'c'
void ex() { char *c = (char *) malloc(100 * sizeof(char)); }


$ clang --analyze main.c
main.c:141:9: warning: Value stored to 'str' during its initialization is
never read
  char *str = (char *)malloc(100 * sizeof(char));
        ^~~   ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
1 warning generated.
```

http://depts.washington.edu/cssuwb/wiki/write_high_quality_c_code

IDE

Bonus

- Visual Studio Code will let you edit files on CSS Linux lab directly
- F2 rename symbol, ALT-SHIFT-F reformat code
- F12 go to definition, ALT-F12 peek at definition
- CTRL-/ comment or uncomment block

Formatting

- Fix your formatting, see https://clang.llvm.org/docs/ClangFormatStyleOptions.html

```
Console:
$ clang-format -style=llvm -dump-config > .clang-format
$ clang-format -i main.c
```