Commonly Uses library Functions

CS-392-A Systems Programming

Instructors:

Georgios Portokalidis – *Stevens Institute of Technology*

string.h: Common String/Array Methods

- **One the most useful libraries available to you**
- **Used heavily in shell/proxy labs**
- **Important usage details regarding arguments:**
  - prefixes: `str` -> strings, `mem` -> arbitrary memory blocks.
  - ensure that all strings are '`/0`' terminated!
  - ensure that `dest` is large enough to store `src`!
  - ensure that `src` actually contains `n` bytes!
  - ensure that `src/dest` don't overlap!

string.h: Common String/Array Methods

- **Copying:**
  - `void* memcpy (void* dest, void* src, size_t n)`: copy n bytes of src into dest, return dest
  - `char* strcpy(char* dest, char* src)`: copy src string into dest, return dest
- **Concatenation:**
  - `char * strcat (char * dest, char* src)`: append copy of src to end of dest, return dest
- **Comparison:**
  - `int strcmp (char * str1, char * str2)`: compare str1, str2 by character (based on ASCII value of each character, then string length), return comparison result
    str1 < str2: -1,
    str1 == str2: 0,
    str1 > str2: 1

string.h: Common String/Array Methods (Continued)

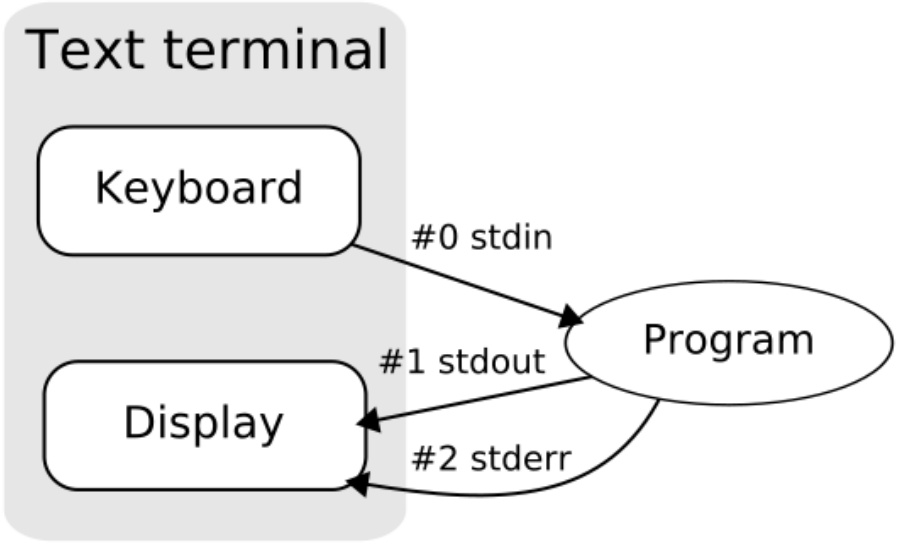
- **Searching:**
  - `char* strstr (char * str1, char * str2)`: return pointer to *first* occurrence of str2 in str1, else NULL
  - `char* strtok (char * str, char * delimiters)`: tokenize `str` according to delimiter characters provided in `delimiters`, return the next token per successive stroke call, using `str = NULL`
- **Other:**
  - `size_t strlen ( const char * str )`: returns length of the string (up to, but not including the '`\0`' character)
  - `void * memset (void* ptr, int val, size_t n )`: set first n bytes of memory block addressed by `ptr` to `val` (use this for *setting bytes only;* don't use to set `int` arrays or anything else!)

stdlib.h: General Purpose Functions

- **Dynamic memory allocation:**
  - `malloc, calloc, free`
- **String conversion:**
  - `int atoi(char* str)`: parse string into integral value (return 0 if not parsed)
- **System Calls:**
  - `void exit(int status)`: terminate calling process, return `status` to parent process
  - `void abort()`: aborts process abnormally
- **Searching/Sorting:**
  - provide array, array size, element size, comparator (function pointer)
  - `bsearch`: returns pointer to matching element in the array
  - `qsort`: sorts the array destructively
- **Integer arithmetic:**
  - `int abs(int n)`: returns absolute value of `n`
- **Types:**
  - `size_t`: unsigned integral type (store size of *any* object)

stdio.h

- **Another really useful library.**
- **Used heavily in cache/shell/proxy labs**
- **Used for:**
  - argument parsing
  - file handling
  - input/output



stdio.h: Common I/O Methods

- `FILE* fopen (char* filename, char* mode)`: open the file with specified filename in specified `mode` (read, write, append, etc), associate it with stream identified by returned file pointer
- `int fscanf (FILE* stream, char* format, ...)`: read data from the `stream`, store it according to the parameter `format` at the memory locations pointed at by additional arguments.
- `int fclose (FILE* stream)`: close the file associated with the `stream`
- `int fprintf (FILE* stream, char* format, ... )`: write the C string pointed at by `format` to the `stream`, using any additional arguments to fill in format specifiers.

Getopt

- Need to include `getopt.h` and `unistd.h` to use
- Used to parse command-line arguments.
- Typically called in a loop to retrieve arguments
- Switch statement used to handle options
  - colon indicates required argument
  - `optarg` is set to value of option argument
- Returns -1 when no more arguments present
- Very useful for Cache lab!

```
int main(int argc, char** argv){
  int opt, x;
  /* looping over arguments */
  while(-1 != (opt = getopt(argc, argv,
"x:"))){
    switch(opt) {
      case 'x':
        x = atoi(optarg);
        break;
      default:
        printf("wrong argument\n");
        break;
      }
    }
}
```

Note about Library Functions

- **These functions can return error codes**
  - `malloc` could fail
  - a file couldn't be opened
  - a string may be incorrectly parsed

- **Remember to check for the error cases and handle the errors accordingly**
  - may have to terminate the program (eg `malloc` fails)
  - may be able to recover (user entered bad input)