Computer Science 126 and 128 **Spring 1996**

Handout # 5: Strings and Files in C: A Cheatsheet

Strings (See K&R pp 93–100, 104–107)

Strings are arrays of characters, terminated by `\0`. You declare strings as you would an array of any type in C, (`char string[N];`). The `\0` character marks the end of the string and provides a convenient way to determine the length of the string.

```
/* returns length of string a */
int strlen(char *a);

/* returns 0 if a == b, negative integer if a < b, positive integer if a > b */
int strcmp(char *a, char *b);

/* same for strcmp, except it only compares first n characters */
int strncmp(char *a, char *b, int n);

/* concatenates string b to end of string a */
void strcat(char *a, char *b);

/* copies string src to string dst */
void strcpy(char *dst, char *src);
```

Files (See K&R, pp 160–162)

In Unix, files are a long sequence of characters in which the last character is a special character called `EOF`. A C program can read characters from a file and write characters to a file. For instance, try the following program. First create a file `mytest.c` in your directory. Then type up this program, compile it, and run it.

```
#include <stdio.h>
void main()
{
  FILE *in, *out;
  int c;

  in = fopen("mytest.c", "r");
  if(in== NULL) return;
  out = fopen("R2D2.c", "w");
  if(out == NULL) return;

  for(c = fgetc(in); c != EOF; c = fgetc(in))
    {
```

```
      fputc(c, stdout); /* put character on screen */
      fputc(c, out); /* put character in R2D2.c */
    }

  fclose(in);
  fclose(out);
  printf("Examine R2D2.c in your emacs editor now!\n");
}
```

Notice that C has a FILE type. Functions that operate on files usually expect an argument whose type is "pointer to file."

```
/* opens a file for reading or writing */
FILE *fopen(char *filename, char *mode);

/* closes file */
int   fclose(FILE *fp);

/* returns next character from file */
int   fgetc(FILE *fp);

/* puts character c into file */
int   fputc(int c, FILE *fp);
```

When a C program executes, the operating system opens two files automatically: `stdin` and `stdout`. If your program prints to the file `stdout`, then that output gets displayed on your screen; this is similar to the `printf` function. Reading from `stdin` has a similar effect to `scanf`.

Note: For more information on any C function, you can type the following at the shell prompt: `man function`, where `function` is the name of the function in question.