CMSC 104
Park , adapted by C Grasso

Strings: Part 2 of 2

Overview

- Turning Strings to Numbers
- Turning Numbers To Strings
- Arrays Of Strings
- Command Line Arguments

Turning Strings to Numbers

- Once we've got a string that contains a character representation of a number in it , we can use the C library to convert it to an integer or a float.
  - `atoi()`
  - `atof()`
- The functions are part of `stdlib.h`
- Both of these functions return 0 if they have a problem

Turning Strings to Numbers

```
char *numString = "3.14";

double pi;
pi = atof (numString);

int i;
i = atoi ("12");
```

Turning Numbers to Strings

- The easiest way to turn a numeric data type value into it's character representation, is to use `sprintf()`

```
int sprintf( char *buffer,
             char *format,
             variables...  );
```

- The `sprintf()` function is just like `printf()`, except that the output is sent to a character array.
- The return value is the number of characters written.

Turning Strings to Numbers

```
char fileName[50];
int  fileYear  = 2012;
int  fileMonth = 01;

for (i=0; i<12; i++) {
    sprintf( fileName, “Monthly%4d-%02d.dat”,
             fileYear, fileMonth );
    fopen(fileName, "w" );
    fileMonth ++;
}
```

Arrays of Strings

```
int main() {

   char *weekday = “Sunday”;

   printf(“Day of the week is %s”, weekday);

   return 0;
}
```

Arrays of Strings

- How is this code different from before?

```
#define WEEKDAYS 7

int main() {

   char *weekdays[WEEKDAYS];

   return 0;
}
```

Arrays of Strings

- What if we initialize all the strings in the array?

```
#define WEEKDAYS 7

char *weekdays[WEEKDAYS] = {
      "Sunday"    ,
      "Monday"    ,
      "Tuesday"   ,
      "Wednesday”,
      "Thursday"  ,
      "Friday"    ,
      "Saturday"
};
```

Arrays of Strings

- What is the easiest way to print out the days of the week?

```
for (i=0; i<WEEKDAYS; i++) {

  printf("Day %d is %s \n",
                  i, weekdays[i]);

};
```

Command line arguments

- You can give input to a C program from the command line.

```
$> gcc  -o prog  prog.c

$> ./prog  10  name1  name2
```

Command line arguments

- These are the official parameters that are sent to main from the OS from the command line.
- Can you guess how the parameters are used?

```
int main ( int argc,  char *argv[] ) {

}
```

Command line arguments

- `main(int argc, char *argv[])`
  - **`int argc`** – gives a count of number of arguments
  - **`char *argv[]`** - defines an array of strings
  - `argv[0]` – program name
  - `argv[1]` to `argv[argc -1]` give the other arguments as strings

Example args.c

```
#include <stdio.h>

main(int argc,char *argv[])
{
   int i;
   for(i=0; i < argc; i++) /*print out args*/

        printf("%s\n", argv[i]);
}
```

Example args.c

```
$> gcc -o args args.c
$> ./args these are my arguments
```

```
./args
these
are
my
arguments
```