More error handling

Recap

- errno.h : declares a global variable errno that stores an integer that encodes errors.
- Many processes automatically write errors to errno if defined.
- Notable example: fopen
- Exit() terminates the calling process and returns a signal.


```
void exit(int status)
```

- EXIT_SUCCESS -> Macro for process exiting successfully.
- EXIT_FAILURE -> Macro for process failing

Example

```c
#include<stdio.h>
#include<errno.h>
#include<stdlib.h>
int main(int argc, char *argv[])
{
  errno=0; //good practice to initialize global variable since it isn't secure.
  if (argc == 1){
    printf("Expect filename\n");
    exit(EXIT_FAILURE); //this returns a signal stating that the process failed to the calling process
  }
  FILE *fptr=fopen(argv[1],"r"); //attempts to open a file whose name is given in the arg stream
  printf("Error number is %d\n",errno);
  perror("error message is");
  if(errno==0){
    exit(EXIT_SUCCESS);//If no error is observed, this immediately terminates the process that calls exit()
    // and sends a signal EXIT_SUCCESS to any process that calls this program
  }
  else{
    exit(EXIT_FAILURE); //as above but returns failure signal
  }

}
```

Assert

- Assert statements are a wonderful way to debug your code.
- The function


```
void assert(int expression);
```


Checks the validity of an expression.

If false it exits the program and gives an error message.

Example

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<assert.h>
int main(int argc, char *argv[])
{
  if (argc ==1){
    printf("please type in at least one index to print\n");
  }
  assert(argc >1); // checks to see if the number of arguments is at least 1 (0th arg is command name)
  char *strLit = "This is a test";
  int x = atoi(argv[1]); //converts input string to integer
  assert( (x>=0) && (x<strlen(strLit)) ); //checks to see if input is within bounds
  printf("character %d of the string %s is %c\n",x,strLit,strLit[x]);
}
```

Setting debug mode

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<assert.h>

#define DEBUG  0

int main(int argc, char *argv[])
{
  if (argc ==1){
    printf("please type in at least one index to print\n");
  }
#ifdef DEBUG
  assert(argc >1); // checks to see if the number of arguments is at least 1 (0th arg is command name)
#endif
  char *strLit = "This is a test";
  int x = atoi(argv[1]); //converts input string to integer
#ifdef DEBUG
  assert( (x>=0) && (x<strlen(strLit)) ); //checks to see if input is within bounds
#endif
  printf("character %d of the string %s is %c\n",x,strLit,strLit[x]);
}
```