

C I/O

CS 241

Data Organization using C

Instructor: **Joel Castellanos**

e-mail: joel@unm.edu

Web: <http://cs.unm.edu/~joel/>

Office: Farris Engineering
Center (FEC) room 321

Lab Instructor: **Dongye Meng**

e-mail: dymeng@cs.unm.edu



3/6/2009

fprintf()

```
int main()
{ FILE *outFilePt = fopen("myFile", "w");
  if (outFilePt == NULL)
  { printf("Error opening file for write\n");
    return 1;
  }
  int n1 = 1;
  int n2 = 1;
  while (n2 < 55)
  { fprintf(outFilePt, "%d ",n2);
    int n3 = n1+n2;
    n1 = n2;
    n2 = n3;
  }
  fclose(outFilePt);
}
```

myFile:
1 2 3 5 8 13 21 34

2 }

scanf()

```
int main()
{ int n1;
  FILE *inFilePt = fopen("myFile", "r");
  if (inFilePt == NULL)
  { printf("Error opening file for read");
    return 1;
  }
  while (fscanf(inFilePt,"%d",&n1) != EOF)
  { printf("%d, ", n1);
  }
  fclose(inFilePt);
}
```

Output: 1, 2, 3, 5, 8, 13, 21, 34,

3

scanf() & gets()

- Textbook: Chapter 7.4
- There is only one thing I really need to say about using scanf() or gets() to read a character string:

Do not do it.



- scanf() and gets() have the exact same problem with memory overrun. You can easily read in more characters than your char* can hold.

4

Reading String Data: Fast & Save

```
#include <stdio.h>
int main()
{
    FILE *inFile = fopen("myDataFile.txt","r");
    int bufferSize = 1024;
    char data[bufferSize];

    //fgets guarantees NULL terminated
    while (fgets(data, bufferSize, inFile))
    { printf("[%s]\n", data);
      }
    return 0;
}
```