

File Handling

Dr. Achraf El Allali

Standard Input and Output

- `scanf`
 - Input from keyboard
- `printf`
 - output to terminal

Opening a File

```
File *fp;
```

```
fp = fopen("file.txt", "r");
```

- fp is the file pointer
- Modes
 - “r” for read mode
 - “w” for write mode
 - “a” for append mode

```
FILE* fopen (char *filename, char *mode);
```

Reading from an open File

- Reading a character from a file
 - `fgetc(fp);`
 - Returns an int (EOF or convert to char)
- Reading a string from a file
 - `fgets(str, 256, fp);`

Reading formatted input

- Read formatted input from a stream
- `int fscanf(FILE *stream, const char *format, ...)`
 - `stream`: input file stream to read from
 - `format`: pointer to a null-terminated character string specifying how to read the input.

Example

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
int main(){
```

```
    char str1[10], str2[10], str3[10];
```

```
    int cNumber;
```

```
    FILE * fp;
```

```
    fp = fopen ("file.txt", "w+");
```

```
    fputs("I love CSC 215", fp);
```

```
    rewind(fp);
```

```
    fscanf(fp, "%s %s %s %d", str1, str2, str3, &cNumber);
```

```
    printf("Read String1 |%s|\n", str1 );
```

```
    printf("Read String2 |%s|\n", str2 );
```

```
    printf("Read String3 |%s|\n", str3 );
```

```
    printf("Read Integer |%d|\n", year );
```

```
    fclose(fp);
```

```
return(0);}
```

Writing to an open File

- Write a character to a file
 - `fputc(c, fp);`
- Write a string to a file
 - `fputs(str, fp);`
 - `int fprintf(FILE *stream, const char *format, ...)`

Writing formatted output

- Sends formatted output to a stream.
- `int fprintf(FILE *stream, const char *format, ...)`
 - `stream`: input file stream to read from
 - `format`: pointer to a null-terminated character string specifying how to write the output.

Example

```
#include <stdio.h>
#include <stdlib.h>
```

```
int main()
{
    FILE * fp;

    fp = fopen ("file.txt", "w+");
    fprintf(fp, "%s %s %s %d", "I", "love", "CSC", 215);

    fclose(fp);

    return(0);
}
```


Example

```
#include <stdio.h>
```

```
#include <stdlib.h>
```

```
main(){
```

```
    FILE * fp;
```

```
    fp = fopen ("file.txt", "w+");
```

```
    fprintf(fp, "%s %s %s %d", "We", "are", "in", 2015);
```

```
    fclose(fp);
```

```
}
```

Example

```
void filecopy(FILE *fpin, FILE *fpout)
{
    int c;
    while ((c = fgetc(fpin)) != EOF)
        fputc(c, fpout);
}
```

Count number of \$ signs

```
#include <stdio.h>
int main ()
{
    FILE * pFile;
    int c;
    int n = 0;
    pFile=fopen ("myfile.txt","r");
    if (pFile == NULL)
        printf("Can't open %s\n","myfile.txt" );
    else{
        do {
            c = fgetc (pFile);
            if (c == '$') n++;
        } while (c != EOF);
        fclose (pFile);
        printf ("The file contains %d dollar sign characters ($).\n",n);
    }
    return 0;
}
```

Error Handling

- Stderr

- Output stream for errors
- Assigned to a program just like stdin and stdout
- Appears on screen even if stdout is redirected

- Exit

- Standard library function
- Terminates the program
- Argument is passed to calling function

Example revisited

```
#include <stdio.h>
int main (){
    FILE * pFile;
    int c;
    int n = 0;
    pFile=fopen ("myfile.txt","r");
    if (pFile == NULL)
        fprintf(stderr, "Can't open %s\n","myfile.txt" );
        exit(1);
    do {
        c = fgetc (pFile);
        if (c == '$') n++;
    } while (c != EOF);
    fclose (pFile);
    printf ("The file contains %d dollar sign characters ($).\n",n);
    return 0;
}
```