File Processing

NOTES

Opening a File and File Modes

```
FILE *infile, *outfile;

infile = fopen("demo1.in", "r");

outfile = fopen("demo1.out", "w");
```

define FILE pointers

open file "demo1.in" for reading

"w" for writing mode

- Note that
  - In "r" mode, file must already exist.
  - In "w" mode, new data will overwrite existing file data (if any).

Opening a File

- We often use **fopen**() function with error checking routine
- If file is opened **successfully**, returns a pointer pointing to the **first data in the file**.
- If file is **failed** to be opened, a **NULL** pointer will be returned.

Week13_Demo1.c

```
#include <stdio.h>
#include <stdlib.h>
int main(void)
{
    FILE *infile;

    if ( (infile = fopen("demo1.in", "r")) == NULL ) {
        printf("Cannot open file demo1.in\n");
        exit(1);
    }

    // process data...
    return 0;
}
```

to use exit()

check whether file is opened successfully

exit function terminates the program immediately

Reading a file

```
float weight, height;
FILE  *fp1, *fp2;

// error checking routine skipped
fp1 = fopen("demo1.in", "r");
fp2 = fopen("demo1.out", "w");

fscanf (fp1, "%f %f", &weight, &height);
fprintf(fp2, "Wt: %f, Ht: %f\n", weight, height);
```

- ***fscanf*** read data from a file.
- ***fprintf*** write data to a file.
- The use of *fscanf* and *fprintf* is similar to *scanf* and *printf*.
  - except for the additional <u>FILE * pointer argument</u>

Reading/Writing a file

What if we are not aware of the total number of the data in the file?

- **fscanf** returns **EOF** if it fails to read any data; otherwise, it <u>returns the number of data that were read and stored</u>.
  - EOF is a macro (constant) whose value is commonly -1.
- Assuming that file contains no error, then a return value of EOF means you have reached the end of the file.

```
FILE *fp;
int  var;

// open file using pointer fp
//while (fscanf(fp, "%d", &val) == 1)

while ( fscanf(fp, "%d", &var) != EOF )
{
    printf("%d\n", var);
}
```

```
Input file:
10 20 30 40
```

```
10
20
30
40
```

Closing a file (Good practice)

```
#include <stdio.h>
int main(void)
{
    FILE *infile, *outfile;
    infile = fopen("demo1.in", "r");
    outfile = fopen("demo1.out", "w");
    // error checking routine skipped

    // data reading and writing skipped here

    fclose(infile);
    fclose(outfile);

    return 0;
}
```

close the file stream represented by this pointer variable