Command-line Arguments

`int main()`

So far we have used the function main() without any argument. But it can take string arguments from the command line when we execute an executable module e.g. `./a.out`. The formal parameters of `main()` may be zero to three.

Example 1

```
#include <stdio.h>
int main(int aNum, char *aList[],
                   char *envL[])
{                        // commLine1.c
    int i;

    printf("Arg. Num.: %d\n", aNum);
    for(i=0;i<aNum;++i) printf("\t%s\n",aList[i]);
    for(i=0; envL[i] != '\0'; ++i)
        printf("%s\n", envL[i]);
    return 0;
}
```

Running

```
$ ./a.out aaaaa bb ccccc dddd eeeee f gg
hhhhhhh
Arg.  Num.:  9
      a.out
      aaaaa
      bb
      ccccc
      dddd
      eeeee
      f
      gg
      hhhhhhh
      SSH_AGENT_PID=2914
      HOSTNAME=goutam
```

Example 2

```
#include <stdio.h>
#include <stdlib.h>
int main(int aNum, char *aList[]) // commLine2.c
{
    int i, n, fact=1;
    if(aNum<2){ printf("No argument\n"); return 0; }
    n = atoi(aList[1]);
    for(i=1; i<=n; ++i) fact *= i;
    printf("%d! = %d\n", n, fact);
    return 0;
}
```

Running

```
$ ./a.out 0
0!  = 1
$ ./a.out 1
1!  = 1
$ ./a.out 5
5!  = 120
```

Example 3

```
#include <stdio.h>
#include <stdlib.h>
#define MAXNO 100
#define ROLL 9
#define NAME 51
struct studData {
        char rollNo[ROLL] ;
        char name[NAME] ;
        float cgpa ;
};
int main(int aNum, char *aList[]) // commLine3.c
{
```

```
int noOfStdnt, i ;
struct studData data[MAXNO] ;
FILE *fpO, *fpI ;

if(aNum < 3) {
   printf("Improper number of arguments\n");
   return 0;
}
fpI = fopen(aList[1], "r");
fpO = fopen(aList[2], "w");
fscanf(fpI, "%d", &noOfStdnt);
for(i=0; i<noOfStdnt; ++i) {
    fscanf(fpI, "%s", data[i].rollNo);
    fscanf(fpI, " %[^0-9]", data[i].name);
```

```
        fscanf(fpI, "%f", &data[i].cgpa);
    }

    for(i=0; i<noOfStdnt; ++i) {
        fprintf(fpO, "%s ", data[i].rollNo);
        fprintf(fpO, " %s", data[i].name);
        fprintf(fpO, " %5.2f", data[i].cgpa);
        putc('\n', fpO);
    }
    fclose(fpI); fclose(fpO);
    return 0;
}
```

Running

```
$ a.out openDat outDat
```