

S&T2024

Computer Programming

(Part 2 – Advanced C Programming Language)

Chapter 1

Lecturer

A/Prof Tay Seng Chuan

E-mail: pvotaysc@nus.edu.sg

Office of the Provost
National University of Singapore

1

Ground Rules

- Switch off your handphone and pager
- No talking while lecture is going on
- No gossiping while the lecture is going on
- Raise your hand if you have question to ask
- Be on time for lecture
- Be on time to come back from the recess break to continue the lecture
- Bring your printed lecture notes to lecture or use a laptop for the e-copy. Do **not** use handphone to read the pdf file.
- Do not open any app, except this pdf file and an editing tool (to take notes).

2

Chapter 1

Command Line Arguments

Consider a program used to encode a text file:

If a character is capital letter, encode it as follows :

Actual Character	A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Encoded Character	K L M N O P Q R S T U V W X Y Z A B C D E F G H I J

If a character is small letter, encode it as follows :

Actual Character	a b c d e f g h i j k l m n o p q r s t u v w x y z
Encoded Character	z y x w v u t s r q p o n m l k j i h g f e d c b a

Other characters remain unchanged. Store the ciphered text (encoded data) in a file. Also, write another program to read the ciphered text and decipher it. Display the plain text on the screen.

We want to make use of this program to encode a text file named monkey.inf, and store the ciphered text in donkey.inf. Here's the test-run result.

Input File (message.inf)

When I call my worker to do something for me, I also give him
some stuff so that he can do the job.
If I ask you to compute the area of a circle, I will have to
give you the radius.

Output File (message.ouf)

Gsvm S xzoo nb dliuvi gl wl hlnvgsrmt uli nv, S zohl trev srn
hlnv hgfuul hl gszz sv xzm wl gsv gly.
Su S zhp blf gl xlnkfgv gsv zivz lu z xrixov, S droo szez gl
trev blf gsv izwrfh.

3

Program

```
/* AC1-1.C */

#include<stdio.h>

main()
{
    FILE *indata,*outdata;
    char this1;

    indata=fopen("message.inf","r");
    outdata=fopen("message.ouf","w");

    while (fscanf(indata,"%c",&this1)==1)
    {
        if (this1>='A' && this1<='Z')
            fprintf(outdata,"%c", (this1 - 2*'A'+'K')%26+ 'A') ;
        else
            if (this1>='a' && this1<='z')
                fprintf(outdata,"%c",'z'-(this1-'a'));
            else
                fprintf(outdata,"%c",this1);
    }

    fclose(indata);
    fclose(outdata);
    return 0;
}
```

Input File (message.inf)

When I call my worker to do something for me, I also give him
some stuff so that he can do the job.
If I ask you to compute the area of a circle, I will have to
give you the radius.

Output File (message.ouf)

Gsvm S xzoo nb dliuvi gl wl hlnvgsrmt uli nv, S zohl trev srn
hlnv hgfuul hl gszz sv xzm wl gsv gly.
Su S zhp blf gl xlnkfgv gsv zivz lu z xrixov, S droo szez gl
trev blf gsv izwrfh.

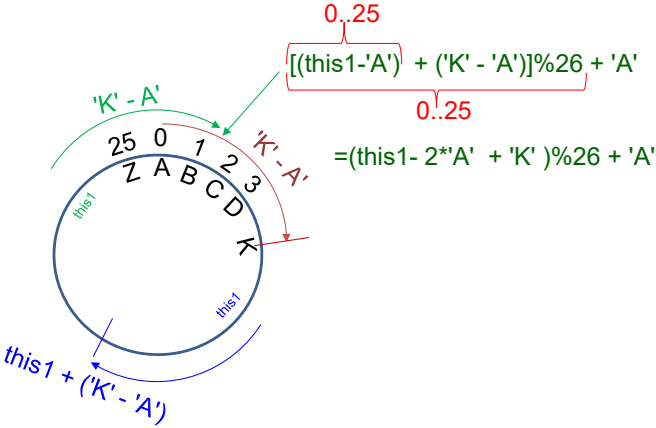
4

Actual Character	A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Encoded Character	K L M N O P Q R S T U V W X Y Z A B C D E F G H I J

```

if (this1>='A' && this1<='Z')
    fprintf(outdata,"%c", (this1 - 2*'A'+'K')%26+ 'A') ;

```

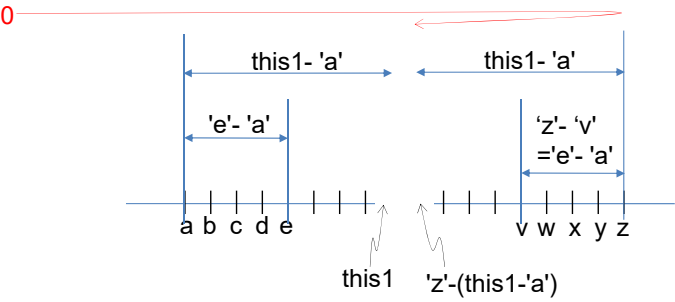


Actual Character	a b c d e f g h i j k l m n o p q r s t u v w x y z
Encoded Character	z y x w v u t s r q p o n m l k j i h g f e d c b a

```

if (this1>='a' && this1<='z')
    fprintf(outdata,"%c",'z'-(this1-'a'));

```

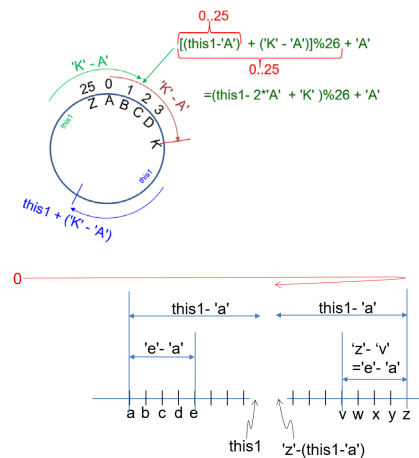


Program

```

/* AC1-1.C */
#include<stdio.h>
main()
{
    FILE *indata,*outdata;
    char this1;
    indata=fopen("message.inf","r");
    outdata=fopen("message.ouf","w");
    while (fscanf(indata,"%c",&this1)==1)
    {
        if (this1>='A' && this1<='Z')
            fprintf(outdata,"%c", (this1 - 2*'A'+'K')%26+ 'A') ;
        else
            if (this1>='a' && this1<='z')
                fprintf(outdata,"%c",'z'-(this1-'a'));
            else
                fprintf(outdata,"%c",this1);
    }
    fclose(indata);
    fclose(outdata);
    return 0;
}

```



Input File (message.inf)

When I call my worker to do something for me, I also give him some stuff so that he can do the job.
If I ask you to compute the area of a circle, I will have to give you the radius.

Output File (message.ouf)

Gsvm S xzoo nb dlipvi gl wl hlnvgsrmt uli nv, S zohl trev srn hlnv hgfuul hl gszg sv xzm wl gsv gly.
Su S zhp blf gl xlnkfgv gsv zivz lu z xrixov, S droo szez gl trev blf gsv izwrfh.

7

What is the Concern (not really a problem)

- Program is not flexible; the input and output filename.ext are all hard-coded.
- Suppose you have 100 messages (stored in 100 separate files) to be encoded before transmission, then you will have to edit and re-compile your program 100 times. By that time most probably your messages are already **out-dated !!!!**

Solution !!!

- *Make the input and output filename.ext as variables.*
- *When you run the program from the command line, also pass in the input and output files.*

8

argc : number of arguments.

argv : an array of string pointers to character string

In the subsequent discussion, filename refers to the descriptor of a file, e.g., a:\monkey.inf or monkey.inf).

Program (flexible version, encode.c)

```
/* AC1-2.C */
```

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

```
main(int argc, char *argv[])
```

```
{
```

```
FILE *indata,*outdata;
```

```
char this1;
```

```
indata=fopen(argv[1],"r");
```

```
outdata=fopen(argv[2],"w");
```

```
while (fscanf(indata,"%c",&this1)==1)
```

```
{
```

```
if (this1>='A' && this1<='Z')
```

```
fprintf(outdata,"%c", (this1-2*'A'+'K')%26+'A');
```

```
else
```

```
if (this1>='a' && this1<='z')
```

```
fprintf(outdata,"%c",'z'-(this1-'a'));
```

```
else
```

```
fprintf(outdata,"%c",this1);
```

```
}
```

```
fclose (indata);
```

```
fclose (outdata);
```

```
return 0;
```

```
}
```

Input File (message.inf)

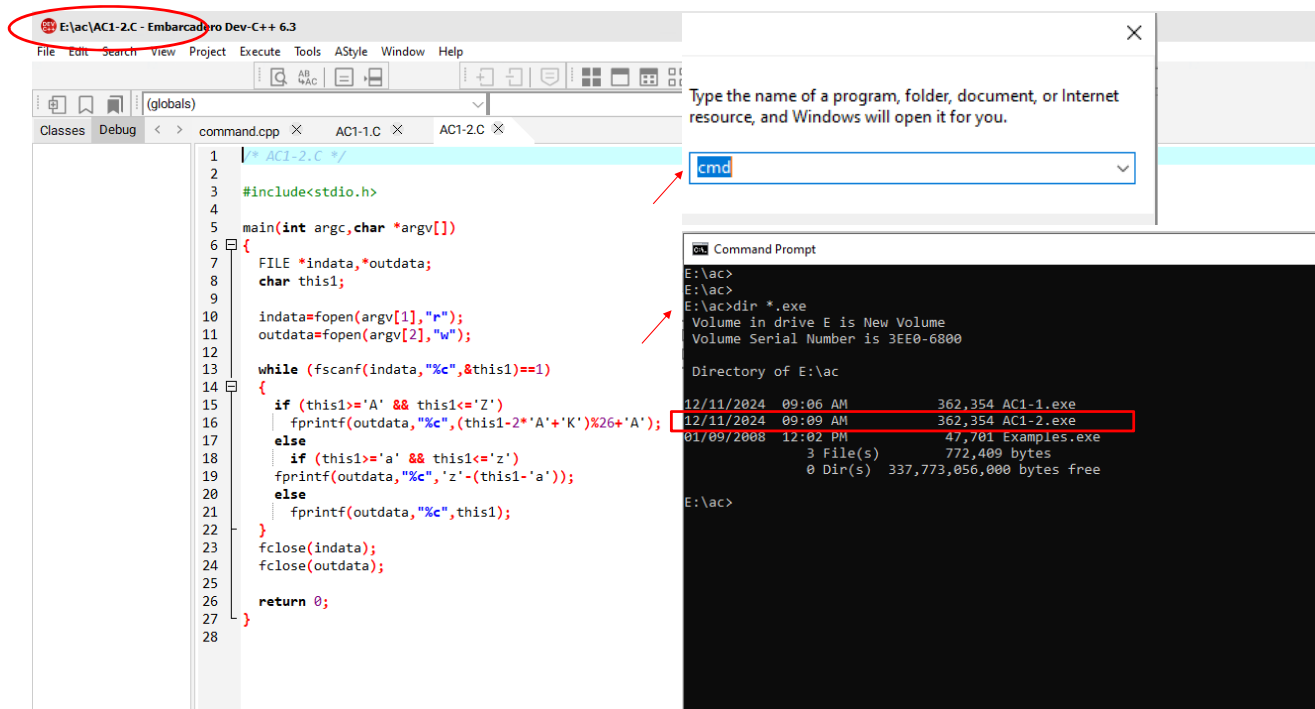
When I call my worker to do something for me, I also give him some stuff so that he can do the job.
If I ask you to compute the area of a circle, I will have to give you the radius.

Output File (message.ouf)

Gvnm S xzoo nb dliipvi gl wl hlnvgsrmt uli nv, S zohl trev srn
hlnv hgfuul hl gszg sv xzm wl gsv gly.
Su S zhp blf gl xlnkfgv gsv zivz lu z xrixov, S droo szez gl
trev blf gsv izwrfh.

Contents of string are entered at the command line

9



```

C:\WINDOWS\system32\cmd.exe
E:\ac>
E:\ac>
E:\ac>dir message.inf
Volume in drive E is New Volume
Volume Serial Number is 3EE0-6800

Directory of E:\ac

11/21/1994  01:12 PM                190 message.inf
               1 File(s)            190 bytes
               0 Dir(s)  337,773,056,000 bytes free

E:\ac>type message.inf
When I call my worker to do something for me, I also give him
some stuff so that he can do the job.
If I ask you to compute the area of a circle, I will have to
give you the radius.

E:\ac>ac1-2 message.inf message.ouf
E:\ac>dir message.*
Volume in drive E is New Volume
Volume Serial Number is 3EE0-6800

Directory of E:\ac

11/21/1994  01:12 PM                190 message.inf
12/11/2024  10:33 AM                190 message.ouf
               2 File(s)             380 bytes
               0 Dir(s)  337,773,056,000 bytes free

E:\ac>type message.ouf
Gsvm S xzoo nb dlipvi gl wl hlnvgsrmt uli nv, S zohl trev srn
hlnv hgfuul hl gszy sv xzm wl gsv qly.
Su S zhp blf gl xlnkfgv gsv zivz lu z xrixov, S droo szez gl
trev blf gsv izwrfh.

```

11

How to use this program ?

- . Edit and compile the program to generate the object code.
- . Run the object code to generate the executable code. If the source file is named encode.c, the default executable code will be saved in encode.exe in the same director of the source file (for DEV C++). Suppose the executable file is named encode.exe. At the prompt sign, use this command to encode a plain file:

c:\> **encode message.inf encoded.ouf**

which means that we run the encode program using monkey.inf (plain text) as an input file, and store the output in donkey.ouf (ciphered text).

If an agent has 100 text files (secret0.inf, secret1.inf, ... , secret99.inf) to be encoded, he can simply use the following instructions with the encode.exe :

```

encode secret0.inf send0.ouf
encode secret1.inf send1.ouf
encode secret2.inf send2.ouf
.
.
encode secret99.inf send99.ouf

```

```

E:\ac>rename ac1-2.exe encode.exe
E:\ac>encode message.inf encoded.ouf
E:\ac>type encoded.ouf
Gsvm S xzoo nb dlipvi gl wl hlnvgsrmt uli nv, S zohl trev srn
hlnv hgfuul hl gszy sv xzm wl gsv qly.
Su S zhp blf gl xlnkfgv gsv zivz lu z xrixov, S droo szez gl
trev blf gsv izwrfh.
E:\ac>exit

```

No recompilation !!!! You cannot afford to compile your program when it is urgent.

So we have addressed the concern.

12

2. The Command-Line Interface

- . In this course we are interested in *argc* and *argv*.
- . Others arguments may be passed as well but we will not discuss them in this course.
- . *argc* is an **int** and contains the number of arguments found.
- . *argv* is **an array of string pointers** to these arguments.
- . The format of a command line instruction is as follows :
progrname arg1 arg2 ... argn
 - progrname is the filename of the executable code.
 - arg1 through argn are optional arguments.
 - Command-line argument is deemed to be a non-white-space character string.
 - Consecutive arguments are separated by multiple spaces and/or horizontal tabs.
 - Arguments are always considered to be **character strings**. (e.g., the integer argument 12345 is interpreted as **string** "12345").

13

Example : [0] [1] [2]

```
C:\> encode   message.inf   encoded.ouf
```

argc = 3 (number of command-line arguments)

argv[0] → "encode\0"

argv[1] → "message.info"

argv[2] → "encoded.ouf\0"

- Take note that argv[0] to argv[2] are pointers to string characters.
- '\0' is a string terminator and is automatically appended to the end.
- '\0' **is one character by itself.**

Example : c:\> encode secret7.inf send7.ouf

argc = 3

argv[0] → "encode\0"

argv[1] → "secret7.info"

argv[2] → "send7.ouf\0"

14

A command-line text processor

```
C:\>textpro infile.inf outfile.ouf /spacing=2 /lm=1 /rm=80
```

```
argc = 6
argv[0] → "textpro.exe"
argv[1] → "infile.inf"
argv[2] → "outfile.ouf"
argv[3] → "/spacing=2"
argv[4] → "/lm=1"
argv[5] → "/rm=80"
```

The advantage of this approach is that the same raw document file can be formatted quite differently just by running textpro with different combinations of command-line arguments. The raw (source) file need not be changed!!

Now we study the flexible version of encode.c again.

15

```
C:\> encode monkey.inf donkey.ouf
```

Program (flexible version)

```
/* AC1-2.C */
#include<stdio.h>
main(int argc, char *argv[])
{
    FILE *indata,*outdata;
    char this1;

    indata=fopen(argv[1],"r");
    outdata=fopen(argv[2],"w");

    while (fscanf(indata,"%c",&this1)==1)
    {
        if (this1>='A' && this1<='Z')
            fprintf(outdata,"%c", (this1-2*'A'+*'K')%26+'A');
        else
            if (this1>='a' && this1<='z')
                fprintf(outdata,"%c",'z'-(this1-'a'));
            else
                fprintf(outdata,"%c",this1);
    }
    fclose(indata);
    fclose(outdata);
    return 0;
}
```

Input File (monkey.inf)

When I call my worker to do something for me, I also give him
some stuff so that he can do the job.
If I ask you to compute the area of a circle, I will have to
give you the radius.

Output File (donkey.ouf)

Gsvm S xzoo nb dlipvi gl wl hlnvgsrmt uli nv, S zohl trev srn
hlnv hgfuul hl gszz sv xzm wl gsv qly.
Su S zhp blf gl xlnkfgv gsv zivz lu z xrixov, S droo szez gl
trev blf gsv izwrth.

In this example I have not made use of **argc**, I will show you the use of **argc** in another program.

encode.c

16

A Case Study

The next program convert a number in base 10 to its equivalent in base b where b is not greater than 8 (for the ease of explanation. You can modify the program to any base value.).


For examples :

Command Line	Response
convert 30 /base=4	132
convert 35 /base=2	100011
convert -13 /base=4	-31
convert 65 /base=7	122
convert 169 /base=9	Base should be less than 9!
convert 13	Wrong Format! Use this instruction : convert dd...d /base=d

Let's do it manually first.

$$30_{(10)} = \underline{132}_{(4)}$$

4	30		
4	7	...	2
4	1	...	3
	0	...	1



17

Program


```

/* AC1-3.C */
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define SIZE 10
main(int argc, char *argv[])
{
    int digit, base, remainder, i;
    char symbol[SIZE];
    if (argc!=3)
    {
        printf("Wrong Format! Use this instruction:\n");
        printf("convert dd...d /base=d\n");
        exit(1);
    }
    if (strlen(argv[2]) != 7)
    {
        printf("Base Format Error!\n");
        exit(1);
    }
    if (*(argv[2]+6) > '8')
    {
        printf("Base should be less than 9!\n");
        exit(1);
    }
}

```

c:\> **convert 30 /base=4**

4	30		
4	7	...	2
4	1	...	3
	0	...	1



18

```


base = *(argv[2]+6) - '0';
digit = atoi(argv[1]);
printf("\t\t");
if (digit < 0)
{
    putchar('-');
    digit = -digit;
}
i=0;
do
{
    remainder = digit%base;
    symbol[i] = remainder + '0';
    i++;
    digit = digit/base;
} while(digit>0);

do
{
    i--;
    putchar(symbol[i]);
} while(i!=0);
putchar('\n');
return 0;
}

```

c:\> **convert 30 /base=4**

4	30		
4	7	...	2
4	1	...	3
	0	...	1




E:\ac>convert 30 /base=4
132

E:\ac>convert 33 /base=2
100001

E:\ac>convert -13 /base=4
-31

E:\ac>convert 100 /base=5
400

E:\ac>convert 65 /base=7
122

E:\ac>convert 65 /base=7
122

E:\ac>convert 169 /base=9
Base should be less than 9!

E:\ac>convert 13
Wrong Format! Use this instruction :
convert dd...d /base=d

E:\ac>exit