This lecture talks about the fourth programming assignment.

This slide shows the basic concept about this assignment. You need to write a program that takes an input file and count how many times each letter appears in the file.

On the slide, the left side shows an example of the input. The right side shows one part of the program’s output. The right side shows that in the input file, the upper case letter M appears twice.

The lower case letter cee appears 7 times in the input.

What are the numbers in front of the letters?

These are called the Ass-Kee code.

A S C I I means the American Standard Code for Information Interchange. The pronunciation of A S C I I is ass kee.

Ass Kee is a table that shows the correspondences between numbers and letters.

Let’s take a closer look of the table.

At the beginning of the table, there are some special symbols. These are not printable letters. At the top, there are different ways to express the numbers, using decimal, hexadecimal, and octal. They use 10, 16, and 8 as the bases. We will talk about different number systems later in this lecture.

The ass kee table includes space and its number is 32. The exclamation mark has number 33. The left parenthesis has number 40. The plus sign has number 43.

The number of the upper case Aee is 65. Upper case Bee is number 66.

The lower case cee is number 99. The lower case n is number 110.

Let’s see how we can use the Ass kee table in a program.

This program uses single quotation marks for a letter. The value of eye starts at the lower case Aee.

As the table in the previous page showed, the value of lower case Aee is 97. Thus, value of eye starts at 97. The print F statement uses eye for two different purposes. If eye is treated as a number, it is 97. If eye is treated as a character, it is the lower case aee.

How does print F distinguish the two purposes of eye? The first one uses % Dee. This treats eye as an integer. The second uses % cee. This treats eye as a character.

The second print F statement prints upper case Aee to upper case F. . They correspond to number 65 to 70.

Next, let’s study how to read one character at a time from a file. Before we read anything from a file, we have to use F open to open a file. F open has two arguments. The first is the file’s name. The second is the mode. If we want to read a file, put lower case R inside the double quotation.

The result of F open is saved in a file pointer called F P T R.

.

If F open fails, F P T R will be NULL.

Why does F open fail? Several reasons may cause F open fail. If the file does not exist, it cannot be read. Thus, F open fails.

It is also possible that the file exists but this programmer has no permission reading the file. In that case, F open can fail.

If F open fails, the program has to handle the problem. Depending on the purpose of the program, the program may ask the user to provide another file, return exit failure, or do something else.

It is important that the program do not F close F P T R because doing so the program’s behavior is undefined.

If F open succeeds, the program may use F get C to read one character. Please notice that the return type is integer, not char. I will explain the reason in the next slide.

Let’s read the documentation of F open and F get Cee.

To find the document, please search Linux and F open on the Internet. F open opens a file and makes it a stream. We will talk about the concept of a stream in the next slide.

If F open succeeds, it returns a FILE pointer. If F open fails, it returns NULL.

Next, let’s read the document of F get C. It takes one argument and it has a stream. This stream is a pointer created by F open.

Please notice that F get C returns an integer. The document says that F get C reads one character and returns it as an unsigned character. Then, the character is cast to an integer. This is necessary because when reading reaches the end of the file, a special symbol called E O F is returned. We will talk about E O F in a moment.

Let’s first understand what a stream means.

You can think of a file as a river. The beginning of the file is upstream and the end of the file is downstream.

There is marker for the current location inside the file. This marker is at the beginning of the file after opening the file using F open. The marker moves toward the end of the file if data is read from or written to the file. You can think of the marker moving down the stream.

How far does the marker move each time? That depends on the size of the data. If a character is read, the marker moves one byte. If an integer is read, the marker moves four bytes. If a double is read, the marker moves eight bytes.

C language has two functions that can report or set the location of the marker. The function F tell returns the current location of the marker from the beginning of the file. The function F seek can set the location of the marker to the beginning of the file, to the end of the file, or somewhere between them. We will see examples of F tell and F seek later.

Let’s go back to the question about E O F. . What is E O F? .

E O F is a symbol defined in the file slash U S R slash include slash S T D I O dot H. .

The G R E P command finds the line where E O F is defined. It is minus one. Please notice that it is not zero.

The slide shows a program that opens a file, reads one character at a time using F get C. . The program prints each character and counts how many characters are read.

The program uses A R G Vee one as the input file’s name. Before using A R G Vee one, we must check whether A R G C is two. If A R G C is one, only A R G Vee zero exists and A R G Vee one does not exist. Thus, if A R G C is not two, the program returns exit failure.

The program uses F open to open the file for reading. If F open fails, F P T R is NULL and the program returns exit failure. Please notice that we should not call F close here.

The program uses F get C to read one character at a time. Each character is printed in both the ass kee value and the character. At the right side, we can see the program’s output.

When the program reaches the end of the file, F get C returns E O F and the program prints the number of characters in the file.

Please be careful not to use unsigned char for C H. .

The reason is that E O F is negative one. If you use unsigned char for C H, C H is always positive and the program will not stop.

Earlier, when we show the ass kee table, at the top we can see decimal, hexadecimal, octal. Now, let’s explain what they are. They represent different number systems.

The most commonly used number system is decimal. The decimal system uses 10 as the base and has 10 digits: 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9.

A commonly used number system is the binary system. This system has only two digits: 0 and 1. The binary system is used widely in computers because electronics can have two states: off and on.

Hexadecimal system is often used. This system uses 16 as the base and has 16 digits. The 16 digits include 0 to 9. Additional 6 digits are needed and they are Aee, B, C, D, E, and F.

Another number system is the octal system. This uses 8 as the base. This system is not used as often as the decimal, binary, or the hexadecimal systems.

Let’s understand how the number systems are used.

I will use subscripts with parentheses to indicate the bases. The first example is one two three four using ten as the base. What does this mean?

We should start from the right end. Four means four times ten with power of zero. Three four means three times ten with power of one, plus four times ten with power of zero.

The entire number one two three four means one times ten with power of three, plus two times ten with power of two, plus three times ten with power of one, plus four times ten with power of zero.

Using the same principle, we can understand the meaning of the binary number one zero one one. It means one times two with power of three, plus zero times two with power of two, plus one times two with power of one, plus one time two with power of zero.

.

The hexadecimal number B 9 C 6 means eleven times sixteen with power of three, plus nine times sixteen with power of two, plus twelve times sixteen with power of one, plus six time sixteen with power of zero.

We can also express numbers that are smaller than the base.

In the first example, a decimal number 5 1 2 point 3 4 means 5 times 10 with power of 2, plus 1 times 10 with power of 1, plus 2 times 10 with power of 0. Point 3 4 means 3 times 10 with power of -1, plus 4 times 10 with power of -2.

Similarly, we can express binary numbers. 1 1 0 point 1 1 means 1 times 2 with power of 2, plus 1 times 2 with power of 1, plus 0 times 2 with power of 0, plus 1 times 2 with power of -1, plus 1 times 2 with power of -2.

The hexadecimal number point Cee 6 means 12 times 16 with power of -1, plus 6 times 16 with power of -2.

The next three examples convert numbers using different bases. The decimal number 534 is the sum of 512 and 22. 22 is the sum of 16 and 6. Thus, decimal number 534 can be converted to hexadecimal number 2 1 6.

16 is the fourth power of 2. Thus, it can be expressed as the binary number 1 0 0 0 0.

The hexadecimal number D is the decimal number 13. To express it as a binary number, it is 1 1 0 1.

Let’s go back to the ass kee table. The @ sign’s ass kee value is decimal 64. It is also hexadecimal 4 0 or octal 1 0 0.

The letter upper case F has ass kee value of decimal number 71. In hexadecimal, it is 4 7.