Basic Types

Chapter 7

(continued)

Characters

- We often need to represent letters, punctuation, and digits in programs.
  - Text to display on the screen or a printed page.
- C provides the *char* type for this purpose.
- A char variable can store a single character
  - a – z
  - A – Z
  - 0 – 9
  - space
  - Various punctuation marks
  - Various “Control” characters
    - tab, linefeed, carriage return, backspace
    - Relics of the age of Teletype® machines

The ASCII Code

- American Standard Code for Information Interchange
- Pronounced ask’ee
- Defines seven bit codes for
  - English letters (upper case and lower case)
  - digits 0 – 9
  - punctuation
  - Control characters
- Values available on line
  - http://www.asciitable.com/ (Or search for ASCII)
- No need to memorize
  - Rarely need numeric values.
  - Look up in table if you do.

char Variables

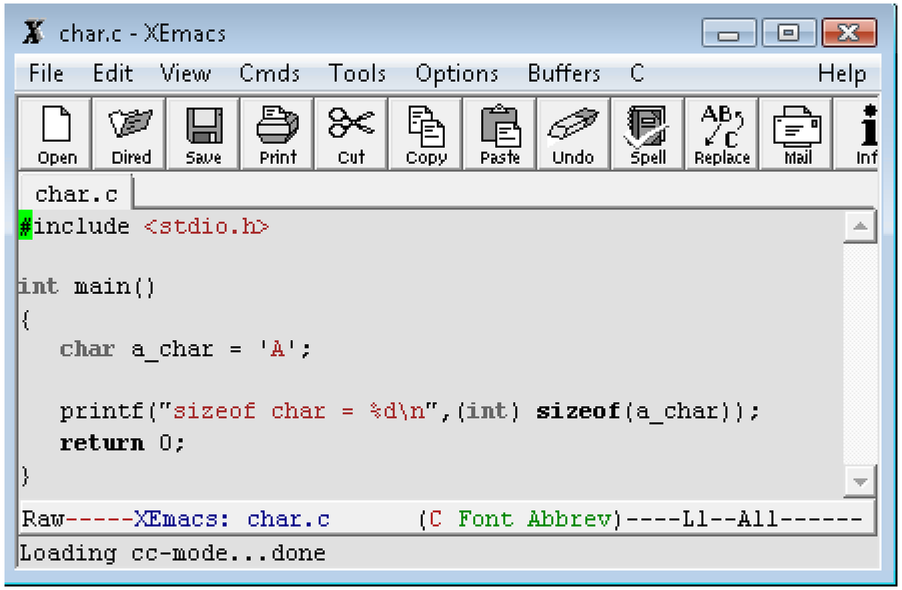
- A variable of type char can hold a single ASCII character.
  - Stored as a single byte. (8 bits)

A char variable

Note single quotes around literal value.

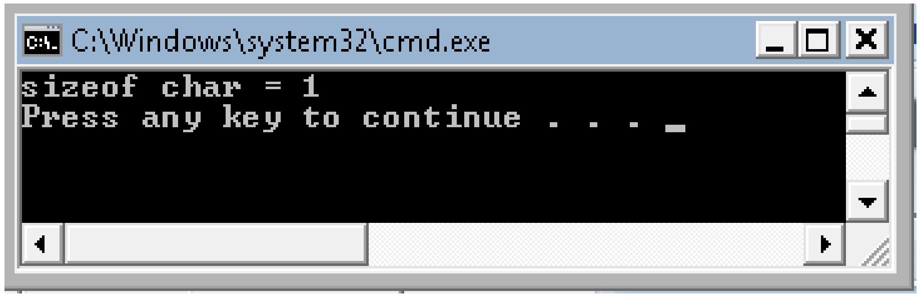
```
#include <stdio.h>

int main ( )
{
    char a_char = 'A';

    printf ("sizeof char = %d\n",
                   (int) sizeof(a_char));
    return 0;
}
```

Program char.c


char.c - XEmacs
File Edit View Cmds Tools Options Buffers C Help
Open Dired Save Print Cut Copy Paste Undo Spell Replace Mail Inf
char.c

```
#include <stdio.h>

int main()
{
   char a_char = 'A';

   printf("sizeof char = %d\n",(int) sizeof(a_char));
   return 0;
}
```

```
Raw-----XEmacs: char.c        (C Font Abbrev)----L1--All------
Loading cc-mode...done
```



```
sizeof char = 1
Press any key to continue . . . _
```

char Variables

- Actually a very short integer.
  - All integer operations can be used.
  - Signed and unsigned versions defined
  - Default is signed.
    - Can vary on other systems.
    - Ignore unsigned for now, like int.

- Can be read from keyboard or written to screen as a character.

char_literal.c

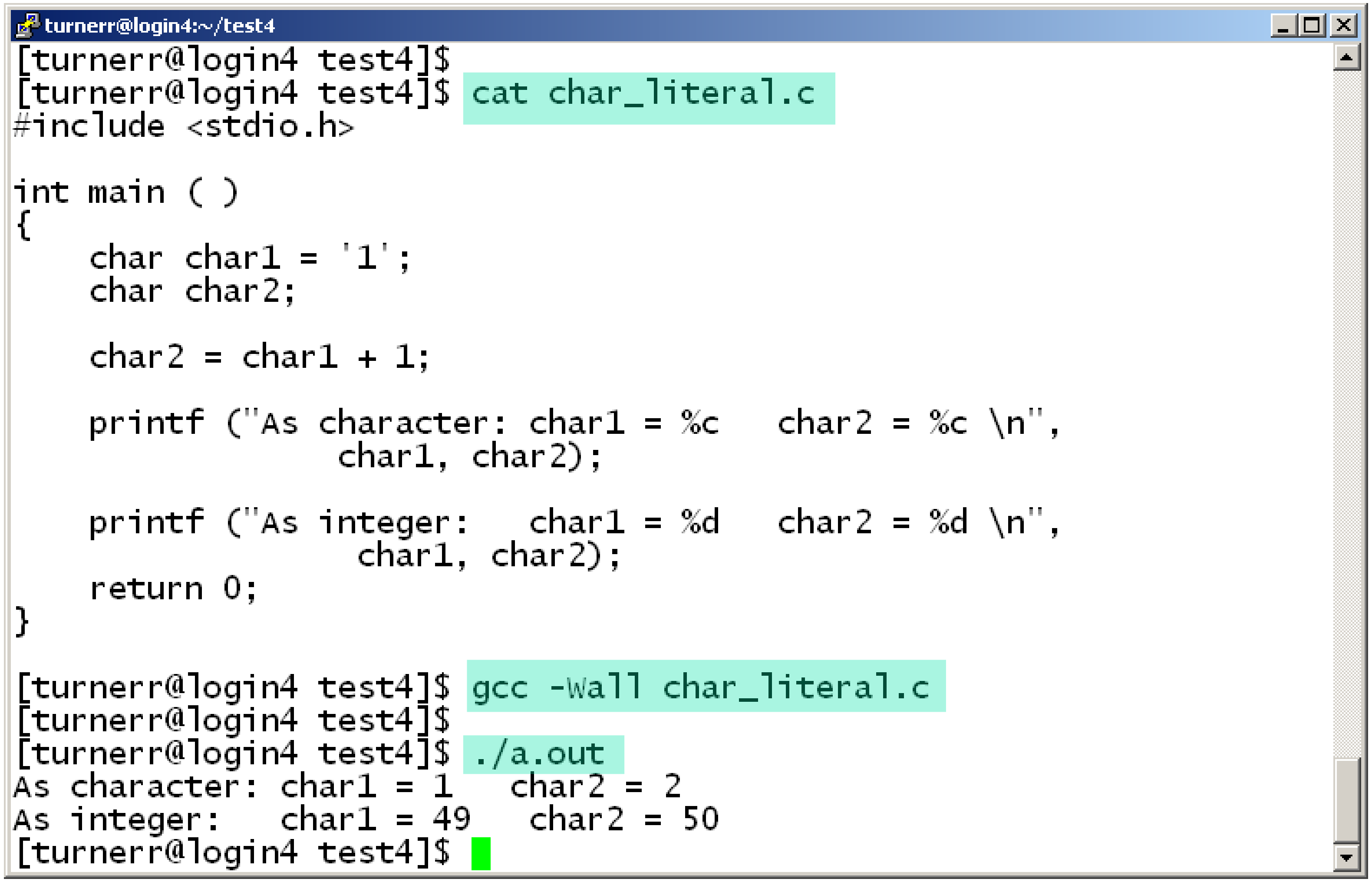
```
#include <stdio.h>

int main ( )
{
    char char1 = '1';
    char char2;

    char2 = char1 + 1;

    printf ("As character: char1 = %c   char2 = %c \n",
      char1, char2);

    printf ("As integer:   char1 = %d   char2 = %d \n",
      char1, char2);
    return 0;
}
```

Set char1 to character literal ‘1’

Output as character

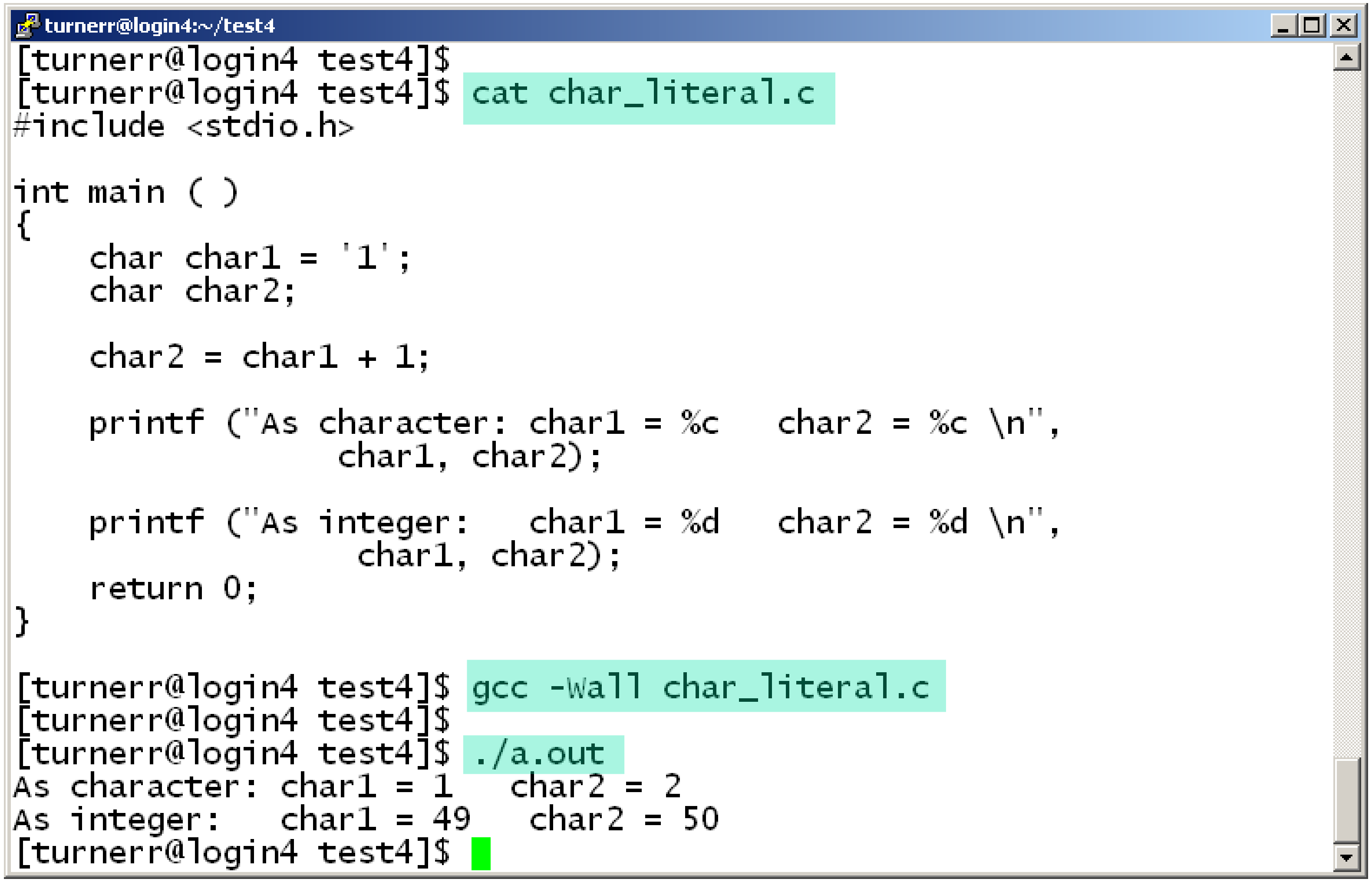
Output as integer

char_literal.c



```
[turnerr@login4 test4]$
[turnerr@login4 test4]$ cat char_literal.c
#include <stdio.h>

int main ( )
{
    char char1 = '1';
    char char2;

    char2 = char1 + 1;

    printf ("As character: char1 = %c   char2 = %c \n",
                 char1, char2);

    printf ("As integer:   char1 = %d   char2 = %d \n",
                  char1, char2);
    return 0;
}

[turnerr@login4 test4]$ gcc -Wall char_literal.c
[turnerr@login4 test4]$
[turnerr@login4 test4]$ ./a.out
As character: char1 = 1   char2 = 2
As integer:   char1 = 49   char2 = 50
[turnerr@login4 test4]$
```

char_literal.c



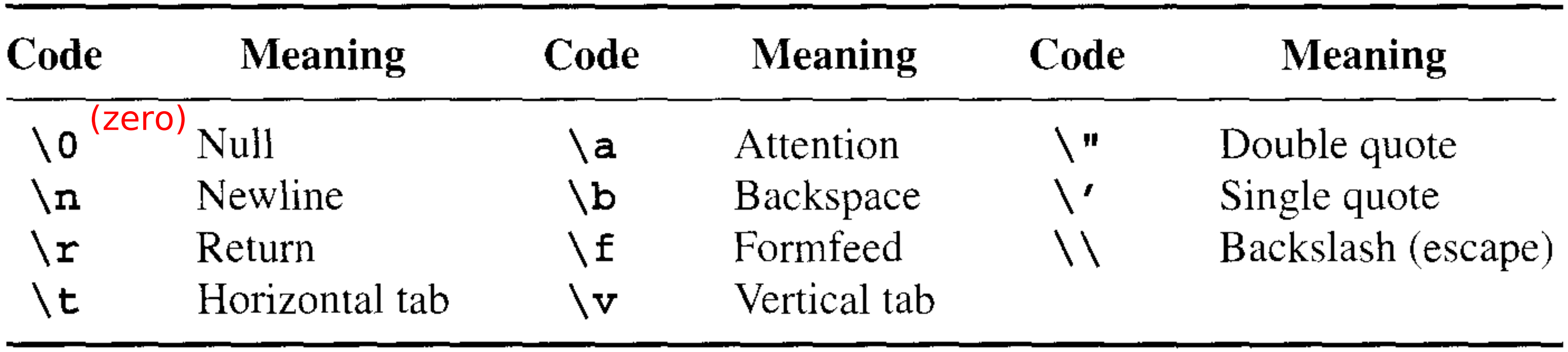
```
[turnerr@login4 test4]$
[turnerr@login4 test4]$ cat char_literal.c
#include <stdio.h>

int main ( )
{
    char char1 = '1';
    char char2;

    char2 = char1 + 1;

    printf ("As character: char1 = %c   char2 = %c \n",
                          char1, char2);

    printf ("As integer:   char1 = %d   char2 = %d \n",
                           char1, char2);
    return 0;
}

[turnerr@login4 test4]$ gcc -Wall char_literal.c
[turnerr@login4 test4]$
[turnerr@login4 test4]$ ./a.out
As character: char1 = 1   char2 = 2
As integer:   char1 = 49   char2 = 50
[turnerr@login4 test4]$
```

Escape Character

- Sometimes we need a character literal that cannot be represented directly in our program.
- Examples:
  - New Line character
  - Null character
  - Tab character
  - Backspace character

Escape Character

- C provides a way to represent these characters in character literals:
  - Backslash followed by a code



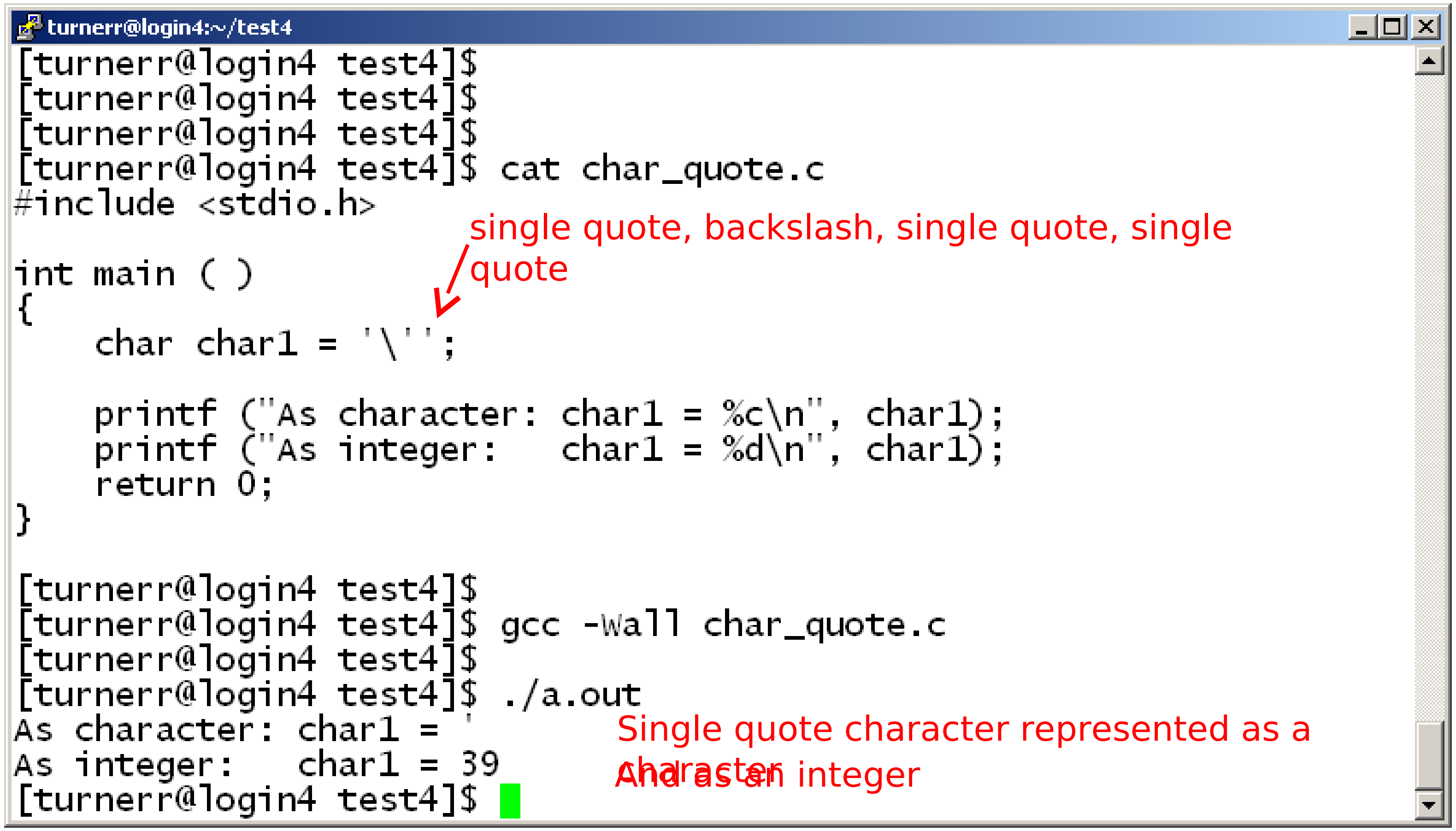
Code	Meaning	Code	Meaning	Code	Meaning
`\0` (zero)	Null	`\a`	Attention	`\"`	Double quote
`\n`	Newline	`\b`	Backspace	`\'`	Single quote
`\r`	Return	`\f`	Formfeed	`\\`	Backslash (escape)
`\t`	Horizontal tab	`\v`	Vertical tab		

Character Literal with Escape Character

```
#include <stdio.h>

int main ( )
{
    char char1 = '\'';

    printf ("As character: char1 = %c\n", char1);
    printf ("As integer:   char1 = %d\n", char1);
    return 0;
}
```

Character literal “single quote”

Character Literal with Escape Character



```
[turnerr@login4 test4]$
[turnerr@login4 test4]$
[turnerr@login4 test4]$
[turnerr@login4 test4]$ cat char_quote.c
#include <stdio.h>

int main ( )
{
    char char1 = '\'';

    printf ("As character: char1 = %c\n", char1);
    printf ("As integer:   char1 = %d\n", char1);
    return 0;
}

[turnerr@login4 test4]$
[turnerr@login4 test4]$ gcc -Wall char_quote.c
[turnerr@login4 test4]$
[turnerr@login4 test4]$ ./a.out
As character: char1 = '
As integer:   char1 = 39
[turnerr@login4 test4]$
```

single quote, backslash, single quote, single quote

Single quote character represented as a character

And as an integer

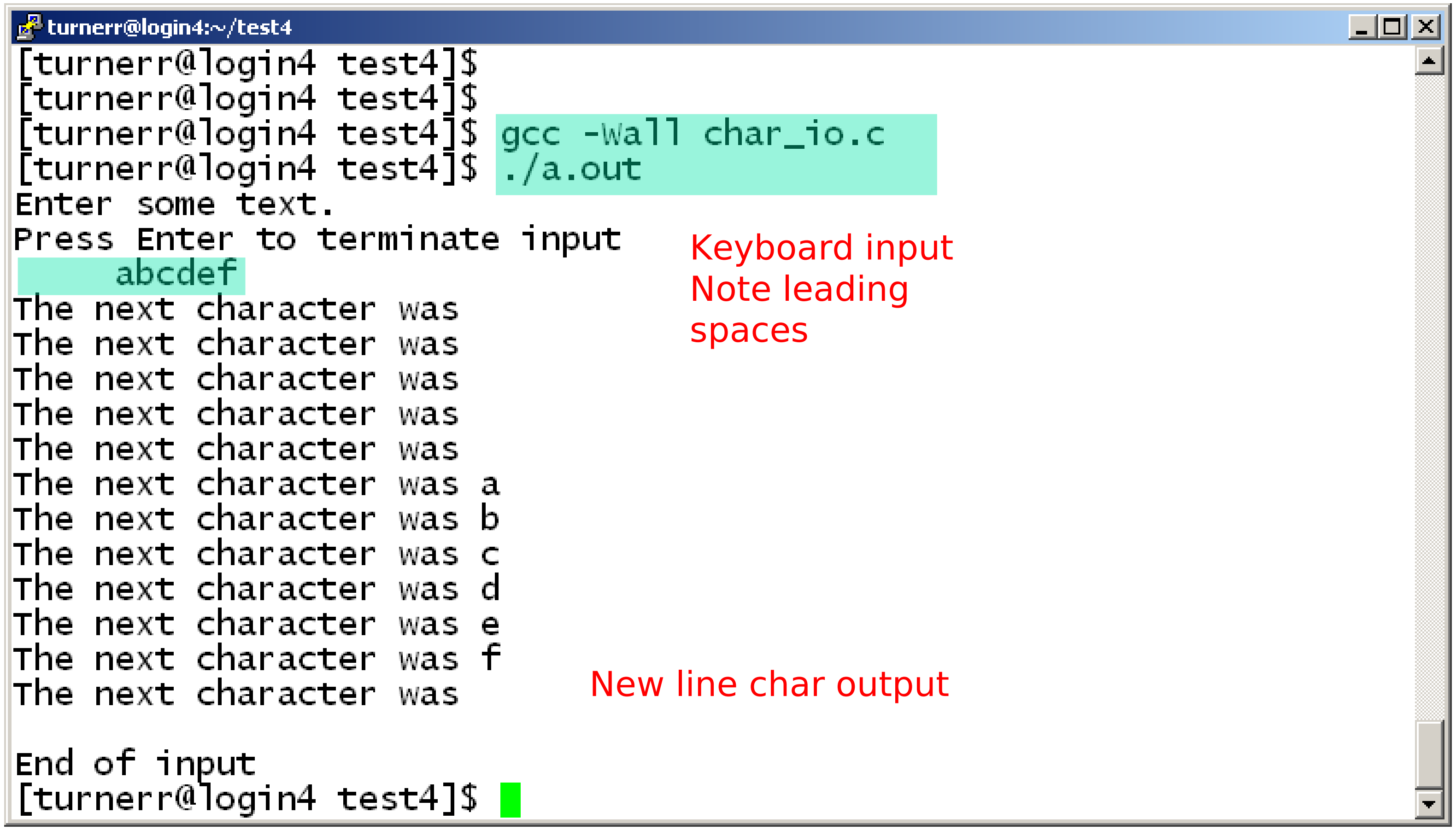
Reading and Writing Characters

- scanf and printf can read and write single characters using “%c”

Reading and Writing Characters

```
#include <stdio.h>
int main ()
{
    char ch = 0;

    printf ("Enter some text. \n");
    printf ("Press Enter to terminate input\n");
    while (ch != '\n')
    {
        scanf("%c", &ch);
        printf ("The next character was %c\n", ch);
    }
    printf ("End of input\n");

    return 0;
}
```

Program Running on Circe



turnerr@login4:~/test4

```
[turnerr@login4 test4]$
[turnerr@login4 test4]$
[turnerr@login4 test4]$ gcc -Wall char_io.c
[turnerr@login4 test4]$ ./a.out
Enter some text.
Press Enter to terminate input
     abcdef
The next character was
The next character was
The next character was
The next character was
The next character was
The next character was a
The next character was b
The next character was c
The next character was d
The next character was e
The next character was f
The next character was

End of input
[turnerr@login4 test4]$
```

Keyboard input
Note leading
spaces

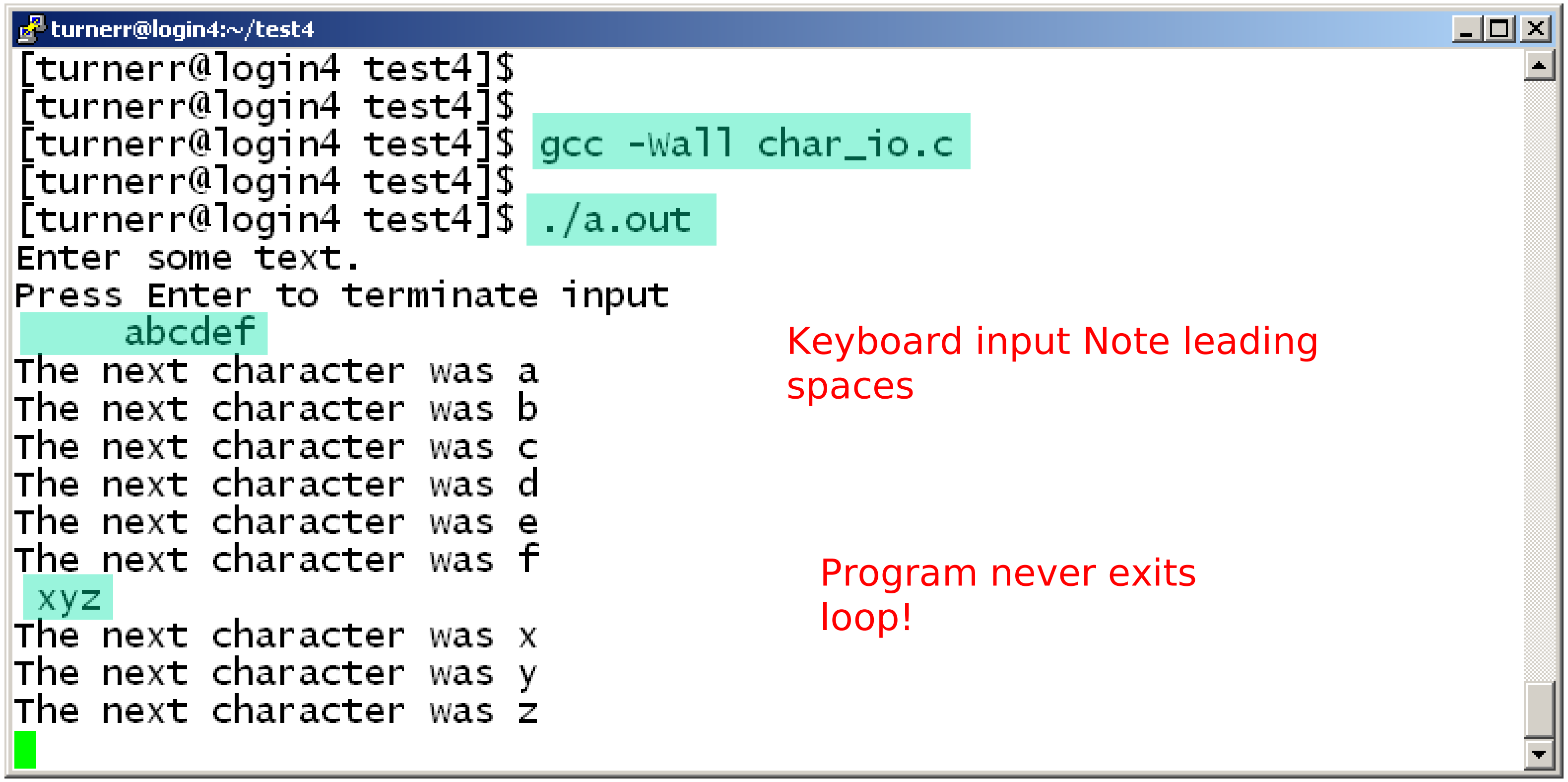
New line char output

Whitespace

- We can tell scanf to skip over leading whitespace by putting a space before the %c

```
scanf(" %c", &ch);
```

Program Running on Circe



```
[turnerr@login4 test4]$
[turnerr@login4 test4]$
[turnerr@login4 test4]$ gcc -Wall char_io.c
[turnerr@login4 test4]$
[turnerr@login4 test4]$ ./a.out
Enter some text.
Press Enter to terminate input
     abcdef
The next character was a
The next character was b
The next character was c
The next character was d
The next character was e
The next character was f
 xyz
The next character was x
The next character was y
The next character was z
```

Keyboard input Note leading spaces

Program never exits loop!

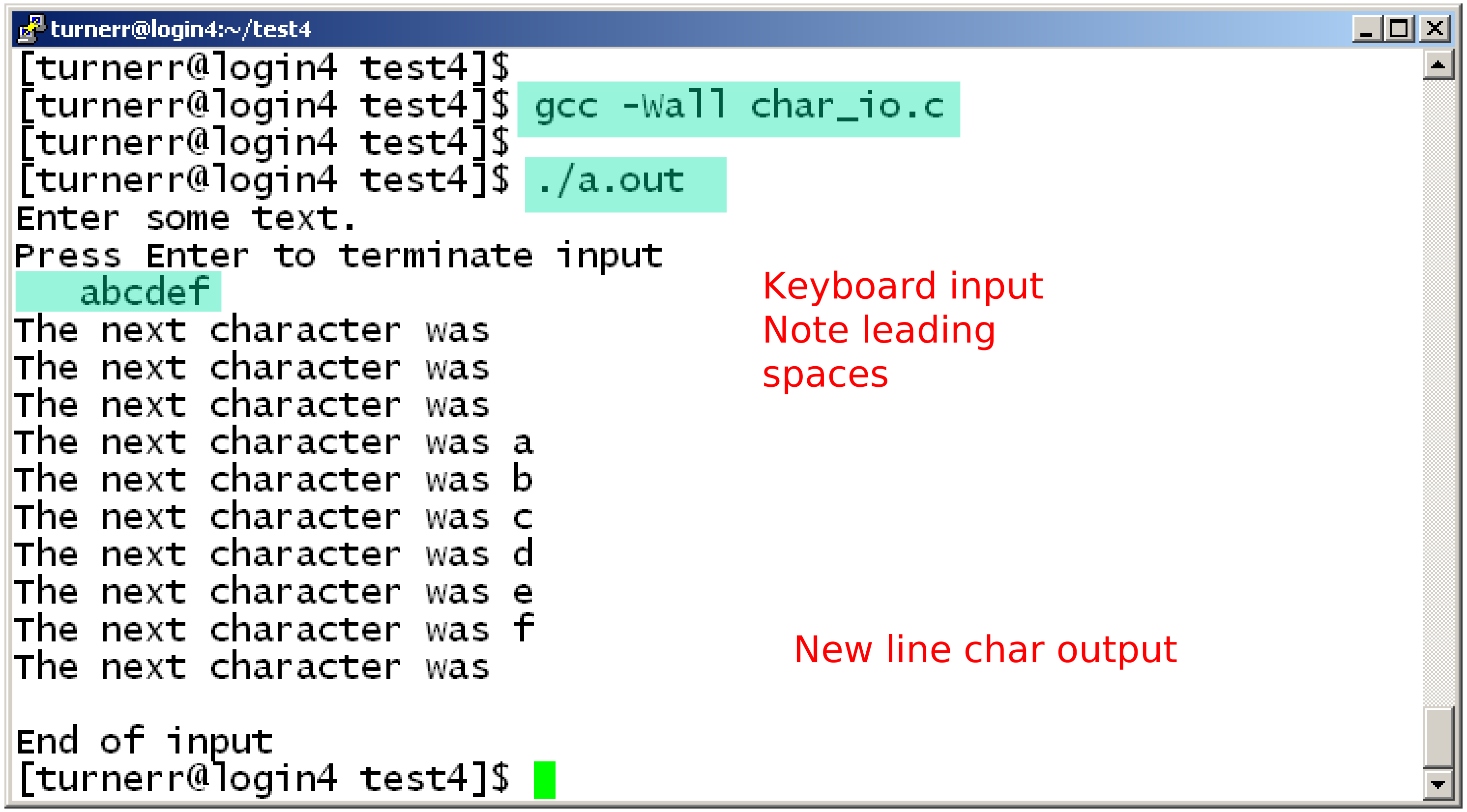
getchar and putchar

- We can also read and write single characters with getchar() and putchar().
- getchar *returns* the character that it reads from the keyboard.
  - Compare to scanf
- putchar() outputs a single character.
- Let’s modify the previous program to use getchar() and putchar()

Using getchar and putchar

```
#include <stdio.h>
int main ()
{
    char ch = 0;

    printf ("Enter some text. \n");
    printf ("Press Enter to terminate input\n");
    while (ch != '\n')
    {
        ch = getchar();
        printf ("The next character was ");
        putchar(ch);
        putchar('\n');
    }
    printf ("End of input\n");

    getchar();
    return 0;
}
```

Program Running



```
[turnerr@login4 test4]$
[turnerr@login4 test4]$ gcc -Wall char_io.c
[turnerr@login4 test4]$
[turnerr@login4 test4]$ ./a.out
Enter some text.
Press Enter to terminate input
   abcdef
The next character was
The next character was
The next character was
The next character was a
The next character was b
The next character was c
The next character was d
The next character was e
The next character was f
The next character was

End of input
[turnerr@login4 test4]$
```

Keyboard input
Note leading
spaces

New line char output

Summary

- Use char to represent text.

- char is really a short integer.
  - One byte.
  - All integer operations apply

- Use %c for char in scanf and printf
- getchar and putchar work with single char variables directly.