

Re: || putchar(ch == '\177' ? '?' : ch | 0100) == EOF)

Re: || putchar(ch == '\177' ? '?' : ch | 0100) == EOF)

Source: http://coding.derkeiler.com/Archive/C_CPP/comp.lang.c/2008-03/msg05343.html

- *From:* Barry Schwarz <schwarzb@xxxxxxxx>
 - *Date:* Sun, 30 Mar 2008 18:12:22 -0700
-

On Sat, 29 Mar 2008 22:36:19 -0700 (PDT), c gordon liddy
<grumpy196884@xxxxxxxxxxxx> wrote:

snip 160 lines of obsolete commentary

Please trim you posts when responding

K&R 7.5 has the text that includes the cat function that is alluded to

snip code you don't intend to use

```
/*filecopy */
void filecopy(FILE *ifp, FILE *ofp)
{
  int c;
  int ch;
  int result;

  while((ch=getc(ifp)) != EOF)
    putc(ch, ofp);
}
```

So, I've got to exchange this for the putc statement:

```
if (iscntrl(ch)) {
  /* ch is a control character */
  int result;

  /*
   * The two characters we want to print. The first is '^';
   * we don't know yet what the second is.
   */
  int ch1 = '^';
  int ch2;
```

Re: || putchar(ch == '\177' ? '?' : ch | 0100) == EOF)

Re: || putchar(ch == '\177' ? '?' : ch | 0100) == EOF)

```
/* Try to print the first character. */  
result = putchar(ch1);
```

The putc call this is replacing used ofp. This is forced to stdout.
Was that deliberate?

```
if (result == EOF) {  
/* Failed, terminate the loop */  
break;
```

Your main calls this function in a loop. A failure here is probably permanent. How do you tell the caller that things are broken and it should stop calling you?

You may want to have the function return a status and let the calling function evaluate that status before iterating the loop.

```
    }  
  
    if (ch == '\177') {  
/* ch is DEL, we want "^?" */  
ch2 = '?';  
    }  
    else {  
/*  
* ch is another control character.  
* Transform 1 to 'A', 2 to 'B', etc. using  
* our intimate knowledge of ASCII encoding.  
*/  
ch2 = ch | 0100;
```

Why limit yourself to ASCII? Why use an octal constant to obfuscate the code? If you want to transform integers to letters, build a static array and select the character using the integer as the index. Make it constant and give it file scope. Something along the lines of
const char transform[] = "@ABCD...XYZ~";
ch2 = transform[ch];
You will need to add a range check on ch since I'm not aware of any guarantee that all control characters have an integer value <= 26.

```
    }  
  
/* Print as above */  
result = putchar(ch2);
```

Re: || putchar(ch == '\177' ? '?' : ch | 0100) == EOF)

Re: || putchar(ch == '\177' ? '?' : ch | 0100) == EOF)

```
if (result == EOF) {  
    break;  
}  
}
```

snip commentary

```
int main(int argc, char **argv)  
{  
  
    FILE *fp;  
    void filecopy(FILE *, FILE *);  
  
    if (argc < 2) printf("die");
```

Avoid portability issues and include a `\n` in your print string.

```
else  
    while (--argc > 0)
```

As a matter of style, the absence of braces will eventually cause you problems. Right now your if and else are close enough to be visually obvious. That will not always be the case. Many adopt the style of using braces even when the range of the loop is a simple one-line statement. I'm not terribly consistent myself in that situation but I always use braces when the range occupies multiple lines.

```
if ((fp = fopen(*++argv, "r")) == NULL)
```

Since you expect the file to contain control character, you should open it for binary input, not text. The reason is that some operating systems will cause `getc` to return EOF when reading the control character that they think marks the end of text.

```
{  
    printf("catv can't open %s\n", *argv);  
    return 1;
```

Use `EXIT_FAILURE` instead of 1 for portability.

Re: || putchar(ch == '\177' ? '?' : ch | 0100) == EOF)

Re: || putchar(ch == '\177' ? '?' : ch | 0100) == EOF)

```
}  
else  
{  
filecopy(fp, stdout);  
fclose(fp);  
}
```

Avoid portability issues and
putchar('\n');
when you finish.

If you run in Windows, a
getchar();
here will keep the window open long enough for you to read the output.

```
return 0;  
}
```

Remove del for email