

Re: Write to file

Source: http://coding.derkeiler.com/Archive/C_CPP/comp.lang.c/2007-06/msg01928.html

- *From:* CBFalconer <cbfalconer@xxxxxxxx>
 - *Date:* Wed, 13 Jun 2007 21:42:33 -0400
-

Ben Bacarisse wrote:

Bill <bill.warner@xxxxxxxxxxxx> writes:

Yes. It has become wacky. What I'm trying to do is read the contents of a file into a buffer, find a particular string in the buffer, make an edit in the found string and write the same file back out.

I suggested on way to go in another sub thread, but let me suggest another which does not suffer from the ugliness of having to read in the whole file.

You can read the file one character at a time. When that character read matches the "next" character in your search string, just update a count of how many matching characters you have seen. If, when you do this, you get to the end of the target string, you know you have found a match and you can print the replacement. When you don't find a match you just reset your count of matched characters.

Try this. It's somewhat simpler (and faster and bufferless).

/*

Leor Zolman wrote:

On 25 Feb 2004 07:34:40 -0800, joan@xxxxxxxx (spike) wrote:

Im trying to write a program that should read through a binary file searching for the character sequence "\name\"

Then it should read the characters following the "\name\" sequence until a NULL character is encountered.

But when my program runs it gets a SIGSEGV (Segmentation violation) signal.

Re: Write to file

Whats wrong? And is there a better way than mine to solve this task (most likely)

I think so. Here's a version I just threw together:

```
*/

/* And heres another throw --- binfsrch.c by CBF */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <assert.h>

/* The difference between a binary and a text file, on read,
is the conversion of end-of-line delimiters. What those
delimiters are does not affect the action. In some cases
the presence of 0x1a EOF markers (MsDos) does.

This is a version of Knuth-Morris-Pratt algorithm. The
point of using this is to avoid any backtracking in file
reading, and thus avoiding any use of buffer arrays.
*/

size_t chrcount; /* debuggery, count of input chars, zeroed */

/* ----- */

/* Almost straight out of Sedgewick */
/* The next array indicates what index in id should next be
compared to the current char. Once the (lgh - 1)th char
has been successfully compared, the id has been found.
The array is formed by comparing id to itself. */
void initnext(int *next, const char *id, int lgh)
{
    int i, j;

    assert(lgh > 0);
    next[0] = -1; i = 0; j = -1;
    while (i < lgh) {
        while ((j >= 0) && (id[i] != id[j])) j = next[j];
        i++; j++;
        next[i] = j;
    }
    #if 0
    for (i = 0; i < lgh; i++)
        printf("id[%d] = '%c' next[%d] = %d\n",
            i, id[i], i, next[i]);
    #endif
}

#endif
```

Re: Write to file

Re: Write to file

```
 } /* initnext */

/* ----- */

/* reads f without rewinding until either EOF or *marker
has been found. Returns EOF if not found. At exit the
last matching char has been read, and no further. */
int kmpffind(const char *marker, int lgh, int *next, FILE *f)
{
int j; /* char position in marker to check */
int ch; /* current char */

assert(lgh > 0);
j = 0;
while ((j < lgh) && (EOF != (ch = getc(f)))) {
chrcount++;
while ((j >= 0) && (ch != marker[j])) j = next[j];
j++;
}
return ch;
} /* kmpffind */

/* ----- */

/* Find marker in f, display following printing chars
up to some non printing character or EOF */
int binsrch(const char *marker, FILE *f)
{
int *next;
int lgh;
int ch;
int items; /* count of markers found */

lgh = strlen(marker);
if (!(next = malloc(lgh * sizeof *next))) {
puts("No memory");
exit(EXIT_FAILURE);
}
else {
initnext(next, marker, lgh);
items = 0;
while (EOF != kmpffind(marker, lgh, next, f)) {
/* found, take appropriate action */
items++;
printf("%d %s : \\", items, marker);
while (isprint(ch = getc(f))) {
chrcount++;
putchar(ch);
}
puts("\\");
if (EOF == ch) break;
}
}
```

Re: Write to file

Re: Write to file

```
else chrcount++;
}
free(next);
return items;
}
} /* binfsrch */

/* ----- */

int main(int argc, char **argv)
{
FILE *f;

f = stdin;
if (3 == argc) {
if (!(f = fopen(argv[2], "rb"))) {
printf("Can't open %s\n", argv[2]);
exit(EXIT_FAILURE);
}
argc--;
}
if (2 != argc) {
puts("Usage: binfsrch name [binaryfile]");
puts(" (file defaults to stdin text mode)");
}
else if (binfsrch(argv[1], f)) {
printf("\'%s\' : found\n", argv[1]);
}
else printf("\'%s\' : not found\n", argv[1]);
printf("%lu chars\n", (unsigned long)chrcount);
return 0;
} /* main binfsrch */
```

--

<http://www.cs.auckland.ac.nz/~pgut001/pubs/vista_cost.txt>

<<http://www.securityfocus.com/columnists/423>>

<<http://www.aaxnet.com/editor/edit043.html>>

<<http://kadaitcha.cx/vista/dogsbreakfast/index.html>>

cbfalconer at mainline dot net

--

Posted via a free Usenet account from <http://www.teranews.com>

.

Re: Write to file