

Re: [newbie] Best way to search for binary data

Source: <http://www.tech-archive.net/Archive/VC/microsoft.public.vc.language/2005-01/0763.html>

From: Jerry Coffin (jcoffin_at_taeus.us)

Date: 01/16/05

Date: Sun, 16 Jan 2005 14:19:06 -0700

In article <63CF5D3C-1C93-4F27-9F1B-670F8323DC59@microsoft.com>, MSalters@discussions.microsoft.com says...

[string searching ...]

> *Actually, Knuth wrote quite a bit about this. E.g. if pattern starts
> with 5 bytes with value 1, and buffer[4] is a 2, you know you don't
> have to check buffer[0] to buffer[3], and the next element you
> should compare is buffer[9]. Highly non-trivial if you're not
> searching random bytes.*

Except in rather specialized circumstances, the Boyer-Moore based algorithms are usually preferred to the Knuth-Morris-Pratt algorithm. The primary advantage of the KMP algorithm is that it never requires any "backwards" processing -- I.e. once you've looked at a particular byte, you're done with it. If you're processing a stream of data and can't store much (or any) of the data during the string match, it works extremely well.

In a situation like this where the data is all in memory, the Boyer-Moore algorithms tend to work better. The one that's usually preferred I've seen referred to as Sunday's variant of Boyer-Moore-Horspool, though I'm not sure Sunday was really the first to invent it -- quite a few people (myself included) seem to have come up with it at close enough to the same times that being sure who did it first would be extremely difficult at best.

Regardless of who originated it, however, the algorithm looks something like this:

```
#include <stddef.h>
#include <string.h>
#include <limits.h>
```

```
static size_t table[UCHAR_MAX + 1];
static size_t len;
static char *findme;
static size_t secondary;
```

```
/*
** Call this first with the string to search FOR.
*/
void init_search(const char *string)
{
    size_t i;

    len = strlen(string);
    for (i = 0; i <= UCHAR_MAX; i++)
        table[i] = len;
    for (i = 0; i < len-1; i++)
        table[(unsigned char)string[i]] = len - i - 1;
    secondary = table[(unsigned char)string[len-1]];
    table[(unsigned char)string[len-1]] = 0;
    findme = (char *)string;
}

/*
** Then call this with a buffer to search IN.
*/
char *strsearch(const char *string)
{
    register size_t shift;
    register size_t pos = len - 1;
    char *here;
    size_t limit= strlen(string);

    while (pos < limit)
    {
        while( pos < limit &&
            (shift = table[(unsigned char)string[pos]]) > 0)
        {
            pos += shift;
        }
        if (0 == shift)
        {
            if (0 == strncmp(findme,
                here = (char *)&string[pos-len+1], len))
            {
                return(here);
            }
            else pos+=secondary;
        }
    }
    return NULL;
}
```

Thanks to Jeff Dunlop, Ray Gardner and Thad Smith, this has fewer bugs than when I originally wrote it. Any bugs that remain are my fault. Thanks to Bob Stout, it's also now easier to use than when I originally wrote it.

microsoft.public.vc.language: Re: [newbie] Best way to search for binary data

Unfortunately, since the last time any of them worked on it, I've done still more rewriting myself, so I may have undone some of the good they did, and re-introduced some bugs. OTOH, this code has been used, studied, and tested for close to 15 years now, so with a little luck most of the problems may be gone...

--

Later,
Jerry.

The universe is a figment of its own imagination.