Sorting strings in C with qsort

2015-03-07

While working on one of the last exercises in Stroustrup's Programming – Principles and Practice Using C++, I came across this little problem: assume you want to sort an array of C-style strings using `qsort()` from the C standard library (`<cstdlib>` in C++ or `<stdlib.h>` in C).

An array of C-style strings looks something like `char* words[ARR_LEN]`, where `ARR_LEN` is the number of strings in your array. `qsort` is declared as follows:

```
void qsort(void* ptr, size_t count, size_t size,
           int (*comp)(const void*, const void*));
```

`ptr` points to your array, `count` is the number of elements in the array, `size` the size of an element and `comp` a comparison function that returns

- a *negative* value if the first argument is smaller than the second,
- a *positive* value if it is bigger than the second, and
- *zero* if they are equal.

Like `strcmp`, if you will.

My strings all have the same maximum length, `WORD_LEN`. Obviously, I try to call qsort like this:

```
qsort(words,ctr,WORD_LEN,strcmp);
```

where `ctr` is the number of strings in `words`, and I want to use `strcmp` as my comparison function. This compiles just fine, but after the call to `qsort`, things are less fine: "stack corrupted". Hmmm.

I realize that the size of an element in my array is not the size of each individual string, but rather the size of a `char*`, because that's what the array is: an array of `char*`, and not an array of actual strings. I change my call to

```
qsort(words,ctr,sizeof(char*),strcmp);
```

Still compiles without problems. Good. And doesn't crash. Even better! However, the results are still not quite what I want. If I start with

```
char* words[ARR_LEN] = { "bb", "cc", "aa" };
```

and sort that, I get `cc aa bb` as the result.

I'm pretty sure I'm not the first person ever to try and sort C-strings with `qsort`, so I google. I come across this Stack Overflow question where somebody had a very similar problem. The problem is that the `comp` argument of `qsort` expects `const void*` arguments whereas `strcmp` expects `const char*`. The solution is to wrap the call to `strcmp` into a function that performs the proper casts, like so:

```
int cmpstr(const void* a, const void* b) {
    const char* aa = (const char*)a;
    const char* bb = (const char*)b;
    return strcmp(aa, bb);
}
```

This takes `const void*`, as required by `qsort`'s `comp`, and casts them to `const char*`, as required by `strcmp`. Neat. I call `qsort` with my new function:

```
qsort(words,ctr,sizeof(char*),cmpstr);
```

Same result! Still no proper sorting.

A closer look at the mentioned Stack Overflow question explains why: the `cmpstr` function provided is not meant for an array of strings, `char* words[]`, but for an array of arrays of characters, `char words[][]`. One less level of indirection!

An element of my array is a `char*`, and the `const void*` in `cmpstr` point to my elements, so what is actually being passed is a `char**`. The proper cast is thus `*(const char**)a`: this points to my string, instead of the pointer to it. The complete function looks like this:

```
int cmpstr(const void* a, const void* b)
{
    const char* aa = *(const char**)a;
    const char* bb = *(const char**)b;
    return strcmp(aa,bb);
}
```

and lo and behold, it works properly!

A more elegant solution can be found on the linux man page for qsort where the casts are in the call:

```
return strcmp(*(char* const*) p1, *(char* const*) p2);
```

“Cast `p1` to a pointer to a constant pointer to a character and dereference that”. How come this wasn’t the first thing that sprung to my mind?