Design Specification – C Program

Design Considerations

Consistency

Where possible, will try to maintain the same structure in the C program as exists in the bash script, such as function and variable names.

Globals

Globals in Bash are the default. In C we may be able to dispense with these, though `#define` statements will still be helpful for the preprocessor.

Use of system(3)

Some of the functions in the bash script (e.g., reading and writing) can be accomplished in C with library functions or system calls, other cannot. For those cases, will experiment with different methods.

- system(3)
- execve(2)
- fork(2)

TODO

- ☒ Update makefile – one for bash script, one for C executable
- ☒ Typedef withing a typedef?
- ☒ Create SERROR?
- ☒ Move processArguments code to main()?
- ☒ Consider `perror()` in `handleError()`
- ☒ Consider globals (e.g., `const char* ETC = "/etc";`) in place of hard-coded strings such as "etc"
- ☒ Move system functions to separate files (system-actions.{c,h})
- ☒ Research use of long switches (e.g., `--help`)
- ☒ Right now the switches are mutually exclusive. Should -fa be allowed?
- ☒ Modify {copy,restore}HostFiles() to be a single function `updateHostFiles(action)` with prep|restore as parameter?
- ☒ Compare costs/benefits of `system(3)` to `exec*` calls
- ☐ Update man page to the new format I discovered (if I can ever locate it again :))
- ☒ Should `handleError()` include `__LINE__` and calling function name?
- ☐ Update bash script with updates from C project
- ☐ Create option to copy updates made to `hosts{,.allow}` files to other systems (mac or linux) somehow (shared dropbox folder?)

Binary executable

- Binary executable located in `~/bin` called `fix-hostfile`

Arguments

- **restore** : restores original hosts file, displays output
- **prep** : creates copy of original hosts file, displays output, calls hblock(1)
- DNS name to add with -a switch

Switches

- -a : add IP entry to allow.list, delete it from hosts
- -f : flush DNS cache and restart the `mDNSResponder` service
- -h : Display usage

main() Parse args and switches, call functions * Handle switches * Handle arguments * Handle actions

void usage(const char *program)

- Display help to user

int updateHostsFiles(const char *src, const char* dst, Action action)

- Modify /etc/hosts
- PREP is essentially cp hosts{,-ORIG}
- RESTORE is the inverse
- if (action == ACTION_PREP), run hblock(1)

int addDnsName(const char *hblock_dir, const char* dns_name, const char *allow_file)

- Add valid DNS name to hblock exception list
- Run hblock(1)

int dnsFlush(void)

- Flush DNS cache
- Restart mDNSResponder daemon
- if action = ACTION_PREP, run hblock(1)

Results

I knew going in that this program was done far better as a bash script. I also knew that the performance of the script *should* be superior to instantiating a binary executable to do the same functions, some of which require `system(3)`.

Therefore, the primary goals of this project was to see how well these same functions could be performed in a C program. While bash scripts are enormously useful, C is nicer to code in – at least for me.

Lessons

1. It's better to have separate folders for each project as VSC does better with this.

- Initially I tried having both the C program and bash executable in the same directory, but this caused complications with both VSC and git.

2. The saying "it's not the writing but the rewriting" is true for coding as well.

- I was surprised to discover things that I missed when creating the bash version of this. In retrospect, these changes should have been self evident.
- For example, I had two functions (`copyHostsFile` and `restoreHostsFile`) in the bash script. Only when writing this is C did it become plainly obvious that these two functions should be in a single function (`updateHostsFiles`).

3. It's good to wait until the code is completed before adding doxygen comments.

- At this point I'm undecided whether I prefer these doc comments in the .c file or the corresponding .h.
- On the one hand, I like the cleaner look of the c files sans api doc comments. It's just cleaner.
- On the other, if these are moved to the .h file, the reader has to bounce back to the header file to see the api doc.
- Also, for the doxygen VSC extension to work in the .h file requires that you explicitly name the variables in the header file; e.g., `void usage(const char *program)` instead of just `void usage(const char *)`. I prefer to not name the arguments in the header file as this becomes a PITA any time I change the corresponding c file argument names.
- A potential solution is to wait until the code is fully baked and then update the declarations in the header file. This would allow putting the api doc comments there.