Spring 2017 CSCI 402

FAQ for Warmup Assignment #1

(Note: this page can change without notice!)

Quick index:

General

Q: Is the "`grading guidelines`" important?

A: According to our fairness policy, we must not grade one student differently from other students. Following the "**`grading guidelines`**" is the **ONLY** way we will grade **all assignments**. The grading guidelines is considered part of the spec for all assignments and you are **expected** to have tested your code against the grading guidelines.

Please note that the grading guidelines says that if you submit an incomplete `README` file, you can lose up to 20 points! Therefore, you must make sure that your `README` file satisfies all the requirements. You must replace every instance of "`?`" in the sample `README` file with your own evaluation.

Q: Can I use Eclipse?

A: I don't think Eclipse runs on Solaris. The information below is for running Eclipse on Ubuntu Linux. Please understand that even if everything runs perfectly under Eclipse, it doesn't count for anything! You have to make sure that your program runs perfectly on `nunki.usc.edu` without Eclispe.

Thanks to Zhiyi Xu, one of our graders in Spring 2014! He wrote a tutorial on how to make Eclipse work with the kernel assignments. You can use the relevant part for the warmup assignments.

A student from a previous semester, Luis Perez Cruz, also mentioned a link about how to use valgrind with Eclipse.

Another student from a previous semester, Kai Lu, also mentioned a link about how to install Valgrind plugin in Eclipse and some useful hints.

Compiler

Q: Is it okay to get a "trigraph" compiler warning?

A: It's an easy thing to get around, so please change your code so that there's no compiler warnings. In this case, you can break the string containing question marks into two string and just concatenate them together. I know this is kind of silly because it's really not the problem with your code but it's the problem with the compiler. But since it's easy to get around, we should all try to make sure that our code gets zero compiler warnings.

Q: Is it okay to get a "`[-Wint-to-pointer-cast]`" compiler warning when compiling "`listtest.c`"?

A: If you are getting the following:

```
listtest.c:80:38: warning: cast to pointer from integer of different size [-Wint-to-pointer-cast]
```

and a bunch of similar compiler warnings on Ubuntu, it's probably because your laptop is a 64-bit machine. If that's the case, don't worry about it since it's warning you that casting a 64-bit pointer to a 32-bit interger may cause problem. Just make sure that you don't get these warning messages on nunki. since nunki is a 32-bit machine.

C-string manipulation

Q: There are lots of C-string manipulation functions. Which are the ones that I need to know?

A: The `weenix` kernel uses the following C-string manipulation functions (they are actually implelmented in the kernel):

```
int    memcmp(const void *cs, const void *ct, size_t count);
void  *memcpy(void *dest, const void *src, size_t count);
int    strncmp(const char *cs, const char *ct, size_t count);
int    strcmp(const char *cs, const char *ct);
char  *strcpy(char *dest, const char *src);
char  *strncpy(char *dest, const char *src, size_t count);
void  *memset(void *s, int c, size_t count);
size_t strnlen(const char *s, size_t count);
size_t strlen(const char *s);
char  *strchr(const char *s, int c);
char  *strrchr(const char *s, int c);
char  *strstr(const char *s1, const char *s2);
char  *strcat(char *dest, const char *src);
char  *strdup(const char *s);
char  *strtok(char *s, const char *d);
```

If these are good enough to implement an operating system, they should be all you need! Do `man` on all these functions to see the man pages for them. The function prototype may be slightly different on other operating systems, but the functionalities should be the same.

File name vs. path name

Q: In the commandline syntax "`warmup1 sort tfile`", is `tfile` a file name or a path? If it's a name then how can we locate the file?

A: If the commandline is `"warmup1 sort xyz"`, `argc` will be 3, `argv[1]` will contain `"sort"`, and `argv[2]` will contain `"xyz"`. Once you verify that `argc` is 3 and `argv[1]` contains `"sort"`, you shouldn't care if `argv[2]` contains a filename or a filepath. Just call something like `open(argv[2])`. If `argv[2]` contains a filename (i.e., `argv[2]` does not contain a `'/'` character), the OS will look for the file in the "current working directory" of the "warmup1" process. The "current working directory" of the "warmup1" process, by default, is the same as the "current working directory" of the "login shell" process on nunki. When you are logged in on nunki, your default "current working directory" is the "home directory" of your nunki account and you can change the "current working directory" using the "cd" command.

Pointer-type declaration

Q: If I declare a list as `My402List *list`, does it mean system will allocate a space with `sizeof(My402List)` to `list`?

A: No. If you do:

```
My402List *list;
```

you are declaring a pointer variable `"list"`. On `nunki`, `"list"` will be 4 bytes long and it's expected to contain a memory location which is the starting address of `sizeof(My402List)` bytes of data. Also, on `nunki`, if you don't initialize `"list"`, it will point to some random memory location which is most likely an invalid memory location. Therefore, if you do:

```
list->num_members;
```

you will be doing the following... Use the content of `"list"` and treat it as a memory location that is the starting address of `sizeof(My402List)` bytes of data. Use the offset that corresponds to `"num_members"` of `My402List` and access that offset from the starting address stored in `"list"`.

In "listtest.c", a list is often declared as:

```
My402List list;
```

In this case, `"list"` is **NOT** a pointer type. `"list"` is a data structure and `"&list"` (i.e., address of `"list"`) is the starting address of `sizeof(My402List)` bytes of data.

By the way, there is no "reference type" in C. So, `"&list"` has nothing to do with a C++ "reference".

Q: I'm still confused! Do I need to allocate memory for `My402List`?

A: There are 2 ways to allocate memory for a My402List.

1) Static allocation. This is the method used in "listtest.c". List data structures are declared as local variable:

```
My402List list;
```

In this case, list (which is a data structure) lives in the stack frame of the function where list is declared. This is also known as an "automatic variable". When the function is entered, memory space is created for this list data structure in the stack frame. When the function returns, the memory space for the list is automatically destroyed (and you must not use it after the function returns). To initialize such as list, you would do:

```
My402ListInit(&list);
```

2) Dynamic allocation. This is NOT done in "listtest.c". Here's what you would do:

```
My402List *pList=(My402List*)malloc(sizeof(My402List));
```

In this case, pList points to a data structure in the dynamic/heap segment. The data structure stays in the heap forever until you free() it. To initialize such a list, you would do:

```
My402ListInit(pList);
```

My402List

"my402list.h"?

A: The first step is to create "`my402list.c`". Since you are given "`listtest.c`" and it's suppose to work, you can just make a copy of it and rename the copy as "`my402list.c`". Then delete all the C code there (variables and functions) and keep only the beginning part of it. Then copy all the function declarations (and **just** the function declarations) from "`my402list.h`" into it. Then remove all the "`extern`" keywords from the function declarations and replace the semicolons from the function declarations with a pair of curly braces (i.e., "`{}`"). Save your new "`my402list.c`" and type "`make`". You should noticed that you will get a bunch of compiler errors because some functions are suppose to return something. Edit "`my402list.c`" and have them return `0` or `NULL` so that the program will compile. Once this is done, you just need to implement each function separately.

When you implement a function, you should first ask yourself, "What is the **precondition** of this function (i.e., what must be true at the beginning of the function if the function is called)?" Other than "`My402ListInit()`", one obvious precondition is that the first argument **must be a valid list**. This means that you must **assume** that if the function is called, the first argument is a valid list and you never have to **check** if the list is a valid list. What if it's not a valid list? Well, it's not your problem! Whoever that's using your list must make sure that it's passing you a valid list Please see my comment about `Unlink()` below.

Now take a piece of paper and draw (by hand) what a valid list look like (for a list containing no elements, a list containing 1 element, a list containing 2 elements, a list containing 3 elements, etc.) You need to check with the warmup #1 spec to make sure what you are drawing is consisten with the spec. For each of the functions, modify the picture to show what the list should look like when the function is about to return. If you need to create a new data structure, then you know what you need to call `malloc()`. (Conversely, if you don't need to create a new data structure, then you also know what you don't need to call `malloc()`!) This would basically be a **postcondition** of the function (i.e., what must be true at the end of the function). Once you know the preconditions and postconditions of a function, all you need to do is to write the code that will take you to satisfy the postconditions for all possible inputs of your function.

Q: The spec for `Unlink(elem)` says that I must not check if `elem` is on the list, what does it mean?

A: `Unlink(elem)` refers to `My402ListUnlink(list,elem)`.

The spec regarding `Unlink()` means that if you must NOT check to see if "`elem`" is on the given "`list`" before you proceed to unlink "`elem`" from the "`list`". The reason is that this can take too long and your list implementation will be too inefficient. For example, if the "`list`" contains one million elements, on the average, it will take you half a million checks to verify that "`elem`" is on the "`list`". That's O(n) behavior where n is the length of the `list`.

The MAIN reason anyonen would use a doubly-linked list is so that insertion and deletion are O(1) operations (i.e., independent of the length of the "`list`"). If you check if "`elem`" is on the "`list`", then the operation cannot be O(1).

Therefore, when you `Unlink()`, you must NOT check if "`elem`" is on the "`list`" or not. What if "`elem`" is NOT on the "`list`"?! Your program may crash! Is that your fault (you, as the list implementor)? Absolutely not! The user of your list (i.e., application programmer) should NEVER call Unlink(elem) with "`elem`" NOT on the "`list`"! It would be your user's fault.

listtest

Q: I got a "unrecoverable error in `RandomShuffle()`" error message. How do I debug?

A: First, you need to read the code of "`listtest.c`" and try to understand under what condition it will print that error message. The next step is to figure out why the condition occur.

Here is a very important debugging trick (well, not much of a trick) I would like all of you to learn... If you look for `64` in "`listtest.c`", you should be able to find that the length of a test list is setup to 64. You should see that a list of 64 integers are created and manipulated. Let's say that something is wrong with your list. As "`listtest`" starts to manipulate your list, things starting to get out of whack. How can you keep track of 64 elements?! That's too much! So, when you are trying to find a bug, change "64" to something small, like 4 or 5. Recompile and re-run listtest and see what's the smallest value you can have and still have your bug to show up. Then debug your code under `gdb`. Set a breakpoint at a place where you know that the list is in a consistent state. Since the list is short, I want you to DRAW your list out on a piece of paper. **Write down the value of EVERY FIELD of every list element.** Then single step through your code and see how these

values change. Hopefully, you will find your bug soon.

To print out a pointer value in `gdb`, you can do something like:

```
printf "0x%08x\n", (unsigned int)(elem->next)
```

`sort` command

Q: I'm done implementing My402List and `listtest` runs perfectly. How do I start doing the 2nd part of the assignment, i.e., implementing the "sort" command?

A: Assming that your implementation of My402List is prefect and `listtest` runs perfectly. Here's what you can do next...

First, rename `listtest.c` as `warmup1.c` by doing the following:

```
mv listtest.c warmup1.c
```

Then use a text editor (such as `vi`, `emacs`, `nano`, etc.) to edit the `Makefile` that you were using and replace every instance of the string "`listtest`" with the string "`warmup1`". At this point, if you type:

```
make
```

An executable file `warmup1` should get created. If you run this program by doing:

```
./warmup1
```

it should not produce any output since this is exactly the old `listtest`. Now you can use a text editor to edit `warmup1.c`, delete most of the code there (since you don't need it for the `sort` command) and change the rest to make it work according to the spec.

Q: The spec says that when you are reading a line of input and if `amount` is ≥ 10 million, it's an error and you should quit the program. The spec also says that when you print, if `amount` is ≥ 10 million, you need to print it in a certain way. Are these two statements in conflict?

A: There is actually no conflict between the two.

If you make sure that no amount is ≥ 10 million, then you should not encounter the case when you print where an amount is ≥ 10 million. Therefore, the "if" part of that particular statement will never be true. Then it doesn't matter what the "then" part of the statement says. So, may be a bit redundant, but still correct.

Please note that even if none of the amount can be ≥ 10 million, the balance can be ≥ 10 million or ≤ -10 million. Therefore, if you use the same routine to print out amount and balance, the ≥ 10 million check becomes relevant.

Q: How do I find out what the current time is?

A: You can use either `time()` (run "`man -s 2 time`" on `nunki`) or `gettimeofday()` (run "`man gettimeofday`" on `nunki`).

`nunki.usc.edu`

Q: My code runs perfectly on my laptop, how do I use `nunki.usc.edu`?

A: To login to `nunki.usc.edu`, you need to use a ssh client. On Windows, you can use `putty` (which can be downloaded from the ITS software site. Of course, the best thing to do is to install Ubuntu 12.04 and `ssh` to `nunki.usc.edu` by simply running the "ssh" command in a Ubuntu terminal. To use `nunki.usc.edu`, you need to be somewhat familiar with Unix. If you are not, please read the tutorial on the class web site.

Once you can `ssh` into `nunki.usc.edu`, you need to copy your submission from your laptop to `nunki.usc.edu`. On Windows, you can use FileZilla. But if you are using `putty`, I would recommend to use `sftp` (which is installed in the same place as `putty`) to "ftp" your files to `nunki.usc.edu`.

Then you need to run through the grading guidelines on `nunki.usc.edu`. You can do this by copying commands from the grading guidelines and paste the commands into your login session on `nunki.usc.edu`.

Once you know that you can get 100 points, you should create a submission file using the command in the spec. Finally, copy the command that corresponds to the section you are in from the spec for running bsubmit and paste the command into your login session on `nunki.usc.edu`.

Q: My code runs perfectly on my laptop, how come it crashes on `nunki.usc.edu`?

A: A major difference between Solaris and Linux is that Solaris does not initialize variables to zero! So, if your code assumes that all uninitialized variables are zero, your code may not work correctly on Solaris. This is why on slide 3 of the Week 1 discussion slides, I asked everyone to **initialize all variables** as a good programming habit!

Q: What does "too many processes" mean on `nunki.usc.edu`?

A: `Nunki.usc.edu` and `aludra.usc.edu` are shared servers. If you run too many processes in parallel, it will not allow you to run more processes.

My guess is that you have too many instances of `FileZilla` running in parallel on `nunki`. When you use `FileZilla`, if you don't disconnect properly, it will still be running on `nunki`!

I think you have to wait until these instances of `FileZilla` timeout and quit by themselves and then you can login again.

In the mean time, you can use `aludra.usc.edu`. Do "**`ps -x`**" to see how many processes on are running and make sure you properly disconnect `FileZilla` (I think you just have to quit `FileZilla`) before you close your laptop.

Q: Why are lines of my `tfile` on `nunki.usc.edu` end with <Cntrl+M>?

A: Such a text file is said to be a **DOS/Windows** text file. Each line in such a text file ends with "`\r\n`". You should convert them into **Unix text files** using the "`dos2unix`" command:

```
mv tfile tfile.dos
dos2unix -437 tfile.dos > tfile
rm tfile.dos
```

Q: I used set locale function to display the amount value directly as a currency. But when I am running the program on nunki, it does not display the output as currency. Does nunki not support the set locale function?

A: I'm pretty sure that nunki supports locale. I don't know why your code is not working on nunki. Please understand that our spec is locale-independent (i.e., even in places where you should use a comma for a decimal point, you are still suppose to use a period for a decimal point). So, if you want to use locale, you should set the locale programatically so that your program works no matter what locale the user/grader is using. This way, you won't be surprised (since you don't have access to the grading account and your program is suppose to work there).

Someone from a previoius semester has suggested the following:

```
setlocale(LC_NUMERIC, "en_US");
```

I have no idea if it works or not. In general, I do not recommend that you write locale-specific code in a CS class! But if you want to mess around with locale, it's your responsibility to make sure that your code works no matter what! In the end, if your submission doesn't work in the grading account on nunki and you end up losing a ton of points, there's nothing I can do about it because you have been warned.

Q: How can I capture all the printout that flew by in a terminal when I run the grading script?

A: In Linux/Unix, there is a command called **`script`** and can create a transcript of everything that gets displayed in a termianl into a file named **`typescript`**. So, you can do the following:

```
script
[ copy and paste part of the grading script here ]
```

```
exit
```

and **typescript** will contain everything that got printed when you run the script.

grading guidelines

Q: What is "usage information"?

A: "Usage" information means "how to use this program". So, it is simply the "commandline syntax" mentioned in the spec. For warmup 1, it's simply (and EXACTLY):

```
warmup1 sort [tfile]
```

So, for any malformed command (where your program prints an error message saying that the command is malformed), ALL you have to do is to add:

```
usage: warmup1 sort [tfile]
```

to satisfy the "MUST give usage information" requirement.

Q: How do I download `$srcdir/f0` to my Ubuntu desktop?

A: You can use the **scp** program, which is just like **cp** except you can copy to and from a remote machine. Let's say you want to download `$srcdir/f0`. First you need to find out what `$srcdir` is. So, you `ssh` to `nunki.usc.edu` and do:

```
set srcdir=~csci570b/public/cs402/warmup1
echo $srcdir
```

It prints:

```
/home/scf-08/csci570b/public/cs402/warmup1
```

On your Ubuntu desktop, you can now start a Terminal program and enter:

```
scp YOURLOGIN@nunki.usc.edu:/home/scf-08/csci570b/public/cs402/warmup1/f0 f0
```

to download `$srcdir/f0` to your current working directory on your Ubuntu desktop. You need to replace "`YOURLOGIN`" with your login ID on `nunki.usc.edu`. If you want to copy a file, say "`xyz`" from your current working directory to your home directory on `nunki.usc.edu`, you can do:

```
scp xyz YOURLOGIN@nunki.usc.edu:
```

Q: `Gdb` says that "`srcdir`" is an "undefined variable", what can I do?

A: You need to replace "`$srcdir` with the full path that it represents. To see what "`$srcdir` represents, do:

```
echo $srcdir
```

Q: Under `gdb`, can I have my program read from the output of "`cat`" (to debug commands used in the grading guidelines)?

A: The first command that uses "`cat`" in the grading guidelines that you will run is:

```
cat $srcdir/f15 | ./warmup1 sort > f15.sort
```

`gdb` does not under "`~csci570b`", so you need to give `gdb` the actual value of "`~csci570b`". According to the above, the actual command that would get run is:

```
cat /home/scf-08/csci570b/public/cs402/warmup1/f15 | ./warmup1 sort > f15.sort
```

By looking at the above command, it seems that `stdin` is just the content of `f15` in `$srcdir`. Then running the above command under `gdb` should look something like:

```
gdb ./warmup1
(gdb) run sort < /home/scf-08/csci570b/public/cs402/warmup1/f15 > f15.sort
```

If you search the Internet, you will find that people say that the above way is the right way to do it. Well, **they are wrong**! (Programming exercise: can you write a program to prove that they are wrong?)

Around slide 61 of the "a simple OS" lecture slides, we have seen the real meaning of "<", i.e., a regular file is "mapped/connected" to file descriptor one.

When you read from the keyboard, you are reading from a "**stream**". One difference between a stream and a regular file is that if you haven't closed a stream, you don't know how long the stream is. For a regular file, you can always find out the file size. Another difference is that with a regular file, you can control where you want to read from (by using something like `lseek()`). If you try to do the same thing with a stream, there is no guarantee that it will work. Therefore, if you have to control where you want to read from and you don't know if you will be reading from a regular file or a stream, don't use functions like `lseek()`.

Anyway, the bottomline is that there is no way to exactly "simulate" the way "`cat`" is used in the grading guidelines when you are debugging under `gdb`! You can redirect input from the file in `$srcdir`, but you need to understand that there is a difference (and you should know **exactly** what the difference is). If you are looking for bugs, you can install `valgrind` and run your program under `valgrind` and hope that it will tell you where your bugs are.

Q: What does "Binary file differ" mean?

A: When you see "Binary file ... differ", it means that "`diff`" is thinking that binary files are being compared and they are different.

My solutions in `$srcdir` are text files, therefore, it's saying that your output file is a binary file! Since you are not trying to generate a binary file, how did you end up generating a binary file? It's because you have a bug in your code and you need to find that bug.

How do you find binary characters in your output (e.g., a `.sort` file)? One way is to do a hexdump of your `.sort` file. On Linux, you can run "`xxd FILENAME`". On Solaris, you can run "`od -x FILENAME`" (or "`~csci551b/bin/hexdump FILENAME`"). Then you look at the output and look for a binary data byte. Well, that may not be so easy.

Another way is to do a hexdump of my solution and do a hexdump of your program output and do a "`diff`" on the two hexdumps. Since the output of the hexdump program is guaranteed to be a text file, "diff" will tell you the first place these two files hexdumps are different. Hopefully, you can pin point the location of your binary character soon after.

Q: What should I do with something like "`Cannot compile : (Comments?)`"?

A: First, look for "Cannot compile" in the grading guidelines. You should see something like:

```
Cannot compile      : -5 to -10, depending on effort to make it work
```

So, your choice is either DEDUCT nothing or something between 5 to 10 points.

The README requirement is that you must replace every "`(Comments?)`" with your own evaluation. So, if you think you won't lose any points (i.e., subtract 0 point), then I would replace the line in the README file by:

```
Cannot compile : -0
```

submission

Q: How can I update my `README` file after I have made a submission?

A: If it's before the submission deadline, you have 2 options.

1. Make a new submission (by running `bsubmit`) and get a new timestamp. The timestamp is what we will use to determine if your submission is early or late and how much extra credit or penalty you will get.

2. Send me modification instructions within 24 hours of the submission deadline. The first 3 lines are free. Then it costs you 3 points per line of modification. Please see:

If you go with (2), we will use the timestamp of your submission (i.e., when you ran `bsubmit`) to determine if your

submission is early or late and how much extra credit or penalty you will get.

So, if you have a lot of changes to your `README` file, (1) would be the way to go. If you have 3 lines or less changes, (2) would be the way to go. If you have more than 3 lines of changes but not a lot of changes, you need to decide for yourself which way you want to go.

Q: Can I find out what I have submitted?

A: Please use the **`bsubmit records`** command mentioned at the bottom of the web page about `bsubmit`.

[*Last updated Thu Jan 26 2017*] [*Please see copyright regarding copying.*]