70

CptS 224 **Midterm Exam** **October 17, 2018**

Name Phong Nguyen ID#: 11557294

This exam contains 21 questions totaling 95 points. Please check that you have all 5 pages. This book is closed book, closed notes, closed laptop, closed neighbor, etc.—do your own work. You should not need any more room than what is provided. However, if you feel you need more, reconsider. If you still feel you need more, then write on the back of one of the pages, and clearly indicate in the space provided that it continues on the back of a page. If you feel one page extra is not enough, consider going to graduate school later in life (just not at WSU please).

Given the following `Makefile`:

```
CC=cc
OBJS= main.o helper.o clients.o
CFLAGS= -o

myBinary: $(OBJS)
        $(CC) -o myBinary $(CFLAGS) $(OBJS)

main.o: main.c myStructs.h
        $(CC) -c $(CFLAGS) main.c

helper.o: helper.c
        $(CC) -c $(CFLAGS) helper.c

clients.o: clients.c
        $(CC) -c $(CFLAGS) clients.c
```

Assume that `myBinary` has already been successfully built.

1. (7 points) What *commands* will be run if the file `myStructs.h` is modified and you type `make` with no arguments?

-2

make would recompile main.o with main.c and myStructs.h since a change has occured. This is due to main.c having the dependency of myStructs.h

2. (7 points) What *targets* will be updated if the file `main.c` is modified and you type `make myBinary`?

-2

Target main.o because it is the only one that was edited. It does this by comparing the timestamps of when the files were last edited.

3. (5 points) Which *target* above would not be built properly if a builtlin rule was used rather than having the above target and command? Hint: think which target has a dependency that a builtin rule would not know about.

✓

main.o → main.c would not be built due to it's dependency on mystructs.h

4. (4 points) What kind of make construct is `OBJS` above called? What is it useful for?

-3 Macro

OBSS calls each of the targets, main.o, helper.o, and clients.o so that they can be run and compiled together rather than calling each one of one by one. It is a way of organizing it all together as well as telling "make", which ones to reccompile.

Given the following listing:

```
drwxr-x---    2 bakken   gridstat      4096 Oct   6 football
drwx------    2 bakken   gridstat      4096 Oct   5 basketball
drwxr-xr-x    2 bakken   gridstat      4096 Oct   3 hockey
drwxrwxr-x    2 bakken   faculty       4096 Oct   9 clubhaus →DIR
drwxr-x--x    2 bakken   faculty       4096 Oct   3 vier
drwxrwxr-x    2 bakken   faculty       4096 Oct   9 bowlingz
-rw-r--r--    1 bakken   gridstat        41 Oct   8 scripts.out
     ↳group ↳others
```

5. ✓ (5 points) Can user `GardnerMinshew`, who is a member of group `athletes`, look at the file `scripts.out`?

Yes

6. ✓ (5 points) Can `GardnerMinshew` modify the file `scripts.out`?

No

7. ✓ (5 points) Can `GardnerMinshew` use the `ls` command to see what is in the `clubhaus` directory?

Yes

8. -1 (5 points) Which directory or directories above could `GardnerMinshew` access a file if given its name, assuming he had appropriate permissions for the file in the directory (not shown), but he could not use ls to find the name of the file? E.g., assuming that file `myFile` existed and in directory DIR he had read access, he could successfully use the command:

```
% wc -l DIR/myFile
```

What value or values of DIR (directories above) would this work for, and `GardnerMinshew` could not do an `ls` command to find the name myFile?

Hockey
clubhaus
bowlingz

Vier

9. -1 (3 points) How would you use the `ls` command to output exactly the following without using the string "`football`" or "`clubhaus`"?

Command: `ls` [fc]* (your options & params go here)

Output: `football clubhaus`

10. -1 (4 points) Give the simplest way to use the `ls` command, list only the items in the current directory whose filename is 4 symbols long? This should not recurse, only list files in the current directory.

ls [a-z][a-z][a-z][a-z]

11. -3 (3 points) Give another command that can output the same as the previous question but is simpler (e.g., no switch/option) and uses the same filename matching pattern

ls [*][*][*][*]

12. -1 (3 points) How would you use the ls command to make a listing similar to that near the top of this page, only not alphabetical but with the files printed oldest first?

ls -u -r

u = print by last used
r = reverse order

13. (4 points) Describe how the `assert` macro works and how you use it in a program, and who its output is intended for.

✓

assert macro is a tool for debuggers/programers, not the user. It is meant for debugging errors in the code without having to use a method like printf, where printf would require you to remove the printf wherever it is no longer needed. For assert, you can simply leave it there and it will not ouput any error codes unless you run into that specific error.

14. (3 points) What advantage does the `TRACE` macro discussed and demonstrated in class have over debugging with just `fprintf` or `printf` directly in your code?

✓

The advantage of trace is that it would help you locate the location of the error in your code rather than you putting a bunch of printf Statements everywhere until you find where the problem is.

15. (3 points) Explain when and how the TRACE macro output is "turned on" (enabled" or "turned off" (disabled). If there is more than one way to do it, which is "best", and why?

-2

TRACE(args)

16. (3 points) What advantage does the `DEBUG` macro package discussed and demonstrated in class have over debugging with the `TRACE` macro?

-2

The DEBUG macro benefits from discovering other errors that TRACE macro normally wouldn't find.

17. (5 points) Give a shell file pattern (for example, for use with `ls`) that would match the English description (below, *legal character* means any symbol that is legal to be in a filename; you should not have to enumerate them):

    The letter 'p', followed by any three legal characters, followed by a decimal digit, followed by zero or more of any legal character, ending with a 'W'.

-2

```
ls    p[a-z][a-z][a-z].*W
ls p[*][*][*].* W
```

18. (5 points) Give an English description of what the shell file pattern `??[aeiou]*t?` would match.

-1

Any files that contain a vowel at the third position and ends with the letter t.

ex) Toast

19. (5 points) Explain what the `mv` command does to the inode of a file and the directory the file is in.

✓

The mv command resets the inode value to that similar of the directory and removes any links previously linked to that file.

20. (5 points) Give the command that would take the output of the `date` command and set the shell variable `var` with it.

-2

```
date | > var
```

0 : stdin
1 : Stdout
2 : stderror

NOTE: Do this problem LAST. It is here for a challenge for someone who got all the rest of the problems done and still has time. It is only worth five points (but a lot of bragging rights…) so unless you are sure your answers above are right it is not worth working on. Two helpers:

- the `-n` flag of echo means a newline will not be printed after the arguments, as it by default is.
- Putting the '`@`' before a command in make means the command will not be printed out if it is executed, but it will execute (and any output from the command of course is printed).

Given the `Makefile` (and assuming all pertinent files exist):

```
TARGETS= hurts unavoidable truth conclusion cold hard

all: $(TARGETS)

truth: cold
        @echo -n "gs dro"
        @touch truth

conclusion: unavoidable
        @echo le
        @touch conclusion

cold: facts
        @echo -n daw
        @touch cold

unavoidable: hard
        @echo -n "gs ru"
        @touch unavoidable

hurts: truth
        @echo ol
        @touch hurts

hard: reality
        @echo -n cou
        @touch hard
```

-n : numbered columns / lines

21. (5 points) What will the following sequence of commands output:

EC
-3

```
$ make > /dev/null
$ touch facts reality
$ make conclusion hurts
```

ol
1. gs ru
1. gs dro
le
1. daw
1. cou