Name: Duohan Xu
Course: CSCI 220 Life Beyond Python
Exam Deadline: December 19, 2022 7:00 p.m.

Part I - Potpourri

POSIX Definitions

1. Definitions:

1.1 **Application Program Interface (API)** - (Sentences within double quotes are from the POSIX directly, and I write my own sentences to conclude after them) “The definition of syntax and semantics for providing computer system services.” It connects computers or software to each other.

1.2 **Built-In Utility** - ”A utility implemented within a shell. The utilities referred to as special built-ins have special qualities. Unless qualified, the term ”built-in” includes the special built-in utilities. Regular built-ins are not required to be actually built into the shell on the implementation, but they do have special command-search qualities.” They are utilities that come with the shell and need no installation once the shell is set up.

1.3 **Executable File** - ”A regular file acceptable as a new process image file by the equivalent of the exec family of functions, and thus usable as one form of a utility. The standard utilities described as compilers can produce executable files, but other unspecified methods of producing executable files may also be provided. The internal format of an executable file is unspecified, but a conforming application cannot assume an executable file is a text file.” It is a file consists of instructions for the OS to perform indicated tasks directly.

1.4 **Protocol** - ”A set of semantic and syntactic rules for exchanging information.” It is a mutually agreed form to regulate the way of information exchanging.

1.5 **Shell** - ”A program that interprets sequences of text input as commands. It may operate on an input stream or it may interactively prompt and read commands from a terminal.” It responds to inputs written in a certain semantic style as commands and let the computer to perform tasks accordingly.

1.6 **Standard Error** - ”An output stream usually intended to be used for diagnostic messages.” It is standardized for visualizing error with messages.

1.7 **Standard Input** - “An input stream usually intended to be used for primary data input.” It is standardized for a program to read the input.

1.8 **Standard Output** - ”An output stream usually intended to be used for primary data output.” It is standardized for a program to write its output.

1.9 **Terminal** - ”A character special file that obeys the specifications of the general terminal interface.” The general terminal interface, like how the API is defined, is an interface with a definition of syntax and semantics to support on communication ports in the system. Terminal is an implementation of these rules.

1.10 **Utility** - ”A program, excluding special built-in utilities provided as part of the Shell Command Language, that can be called by name from a shell to perform a specific task, or related set of tasks.” ”A utility program shall be either an executable file, such as might be produced by a compiler or linker system from computer source code, or a file of shell source code, directly interpreted by the shell. The program may have been produced by the user, provided by the system implementor, or acquired from an independent distributor.” Utilities are programs that can perform a task by calling its name in the shell.

Git

2. gitignore link

Tarball

3. -

Part II - Bash

Learning the Shell

4. -

5. Types of commands:

Type	Example
Executable program	/a.out (produced by compiling a C file)
Command built into the shell itself	type ls
Shell function	myfunc () { echo ”a” }
Alias	alias ”l=’ls -l’

6. Redirection operators:

Redirection	Operator
Redirecting input	<
Redirecting output	>
Redirecting error	2>
Appending redirected output	>>
Redirecting standard output and standard error	&>
Appending standard output and standard error	&>>

7. Types of expansion:

    1.1 Pathname Expansion

    1.2 Tilde Expansion

    1.3 Arithmetic Expansion

    1.4 Brace Expansion

    1.5 Parameter Expansion

    1.6 Command Substitution

8. Types of quoting:

Type	What does it suppress?
Double Quotes	Special characters in the text (except $, \, and `)
Single Quotes	All expansions inside the quotes
Escaping Characters	A single character quoted by the backslash (often inside double quotes)

9. Types of files:

Name	Attribute	Type
file1	-	A regular file
file2	l	A symbolic link pointing to /dev/loop0
file3	l	A symbolic link pointing to /dev/null
file4	d	A directory
file5	l	A symbolic link pointing to file5
file6	p	A named pipe
file7	l	A symbolic link pointing to /var/run/nvidia-xdriver-106e9220

10. The nine characters in file attributes (given by long listing) after the first character (file type) are called the file mode. It can be divided into three groups, each with three characters representing the read, write and execute permissions. The first group in sequence is the permission information for the file's owner, second for the file's group owner, and third for everybody else. If the object is a file, the first character "r" in one group indicates whether the file can be opened and read; the second character "w" indicates whether the file is allowed to be written to or truncated (but not renamed or deleted as these are regulated by the attributes of its directory); and the third character "x" indicates whether the file can be treated as a program and executed. If the object is a directory, "r" shows whether its contents can be listed if the execute attribute is also set; "w" shows whether its files can be created, deleted, and renamed if the execute attribute is also set; "x" allows a directory to be entered. If any of these characters are replaced by "-" in the file mode, it means that the corresponding permission for the certain group is not given. For example, a file mode "rwx- - - - - -" for a file means that only its owner can read, write and execute the file.

11. It means that the object can be read, written, and executed by its owner, but only readable for the group and any other users.

12. It means that the object can be executed by anyone, but it is only readable and writable for its owner.

13. It means that the object can be read and executed by anyone, but it is only writable for its owner.

14. It means that the object can only be read and written (but not executed) by its owner; anyone else should have no permission at all.

15. It means that the object can be read, written and executed by anyone.

16. Because the file mode for ˜mgwhite/public_html/lbp/FINAL/dir1 is ”rwxr- -r- -”, which means that only the permission for listing is open to the group and world users. Except the owner himself, one can only see what is in the directory but have no access to anything inside.

17. Because the file mode for ˜mgwhite/puiblic_html/lbp/FINAL/dir2 is ”rwx- -x- -x”, which means that only the permission for entering and open files in this directory is open to the group and world users. Except the owner himself, one can only directly open file.txt inside (if the file itself also allows users to read it) but can not see the listing for this directory. Moreover, the file mode of file.txt is ”rw-r- -r- -”, which means that everyone can open and read this file.

18. These commands work for student accounts. For wall, it has a file mode ”rwxr - sr - x”. We are not in its group ”tty”, so we need to look into the world group in which all users can read and execute this executable. For write, it is a soft link pointing to an ultimate location /usr/bin/bsd-write that has the same file mode as wall. As a symbolic link, write has dummy permissions allowing anything, and we have to look into the the permission for the location it points to (and it has the same permission as wall, enabling student accounts to use these two commands as accounts are not in its group ”root”).

Configuration and the Environment

19. Types of data:

    1.1 Environment variables

    1.2 Shell variables

20. It lists directories that are searched when entering the name of an executable program. The system can search each directory inside this variable and try to find the target executable program inside directories searched. Therefore, one does not need to write the whole path to the executable if its directory is listed in the $ PATH variable.

21. The environment variable is LD_LIBRARY_PATH.

22. Types of sessions:

    1.1 Login shell session

    1.2 Non-login shell session

23. Startup files:

    1.1 For login shell sessions: /etc/profile, ~/.bash_profile, ~/.bash_login, ~/.profile .

    1.2 For non-login shell sessions: /etc/bash.bashrc, ~/.bashrc .

Common Tasks and Essential Tools

24. *POSIX Basic* recoginzes ^, $, ., [,], and as metacharacters; *Extended Regular Expressions* adds (,), {, }, ?, +, and | as metacharacters on the basis of *POSIC Basic*. However, the (,), {, and} characters are treated as metacharacters in BRE if they are escaped with a backslash, whereas with ERE, preceding any metacharacter with a backslash will make it be treated as a literal. We have to use a different grep (egrep or grep -E) to support ERE.

Part III - C

Maintain your perceptron ADT

25. Valgrind is a program for debugging and profiling Linux executable files. It helps programmers to resolve memory leak issues in the target file.

26. An error occurs every time when new_ Data or new_ Model is called. This is resulted by incorrectly freeing the memory after a Data or Model instance is initialized. Simply freeing data and model in main cannot solve this issue since the pointers inside the structure data or model is pointing to are not freed by this action, causing memory leaks. Also, I allocate the space for Data instance using the size of Data and Model instance using the size of Model, which is incorrect; this will cause invalid writing and reading since not enough space is allocated for the structure. To solve this issue, I need to free every pointer inside the structure, and modify malloc methods used in perceptron.c to allocate the right amount of space.

27. -

28. -

Evolve your perceptron ADT

29. 80%

Part IV - Python

Evolve a simple interpreter

30. -

Part V - Java

Describe the functionality of a class

31. Finished (1), please look at the Java files in the tarball.