Assignment 1

CS962: Operating System Principles

January 30, 2023

1 Infinite Chain of Unary Operations [40 Marks]

In this question, you need to write three c programs defined in `Part1/square.c`, `Part1/double.c` and `Part1/root.c` which perform square, double and square root operations respectively on a non-negative integer such that generated executables with these programs can be chained in any pattern.
The order of the operations in the chained pattern would be from **left to right**.

Synopsis

```
$ ./square root double root square integer_number
```

Example

```
$ ./square root double root 8
4
```

Since, the order of the operations in the chained pattern is from **left** to **right** hence, the chained pattern should be viewed as:

```
    root(double(root(square(8)))) = 4
```

Output

Print the final result only (as shown in the example).

Note

- Atleast 1 unary operation and atmost 100 unary operations will be specified during testing.
- If your implementation results in parent-child relationship between processes, then parent process must wait for its child process to exit.
- You can assume that the result of operations will always fit in unsigned long long data type.
- If square root of number is a fraction then round off the result to the nearest integer, e.g. 2.5 should be rounded to 3 and 2.4 should be rounded to 2. This needs to be done every time square root is performed. For example: Answer for chain of operations done below will be 8 and not 6

```
    ./root square root square root square double 3
```

Error handling

In case of any error print "UNABLE TO EXECUTE" as output.

System calls and library functions

You **must only use** the below mentioned APIs to implement this question.

```
    - fork                  - malloc
    - exec* family          - free
    - str* family           - exit
    - ato* family           - wait/ waitpid
    - printf, sprintf       - sqrt
    - round
```

Testing

Run the script `Part1/run_tests.sh` for running the provided sample test cases which contains 3 test cases. A sample output after running the script would be:
Test 1 is Passed
Test 2 is Passed
Test 3 is Passed

2 Simple Tar Utility [60 Marks]

GNU tar is an archiving program designed to store multiple files in a single file (an archive), and to manipulate such archives. You can read more about it in its man pages ($man tar, $info tar). You can use original tar utility to familiarize yourself with it.
In this question you will be implementing a simpler version of tar utility that should support creation of tar file, extraction of all or a single file from a tar file, listing the contents of tar file. Detailed description of your tasks is as following:

2.1 Creation and Extraction of tar file [40 Marks]

2.1.1 Creation of tar file

Command line arguments

```
    $./myTar -c <path of a directory> <tar filename>
Eg: $./myTar -c ~/Downloads/Movies   MyMoviesCollection.tar
```

Description of command line arguments

- myTar: simple tar utility that you have to implement
- -c: This argument tells your tar utility that creation of tar file operation needs to be done.
- path of a directory: path of directory whose all files need to be archived. You can assume that this directory will contain only regular files. Name of each regular file will be of max 16 characters long. There will atleast 1 file in this directory and atmost 30 thousand files. Size of each file can vary from 0 bytes to 1GB. You can assume that all paths passed to your utility will be absolute paths in all subparts of this question.
- tar filename: Name of the tar file that will be created by your utility as output file. It should be created in same directory whose file's are to be archived. Example: In figure 1, MyMoviesCollection.tar gets created in ∼/Downloads/Movies directory.

Description of functionality

- Your tar utility should read the contents of input directory to know the name of each file to be archived. You can use readdir() for this purpose. A sample program to demonstrate the opening, reading, closing of a directory is present in `Part 2` directory (`Part2/example_program_for_dir_ops.c`).Please refer the manpages of opendir(), readdir(), closedir() for more info).

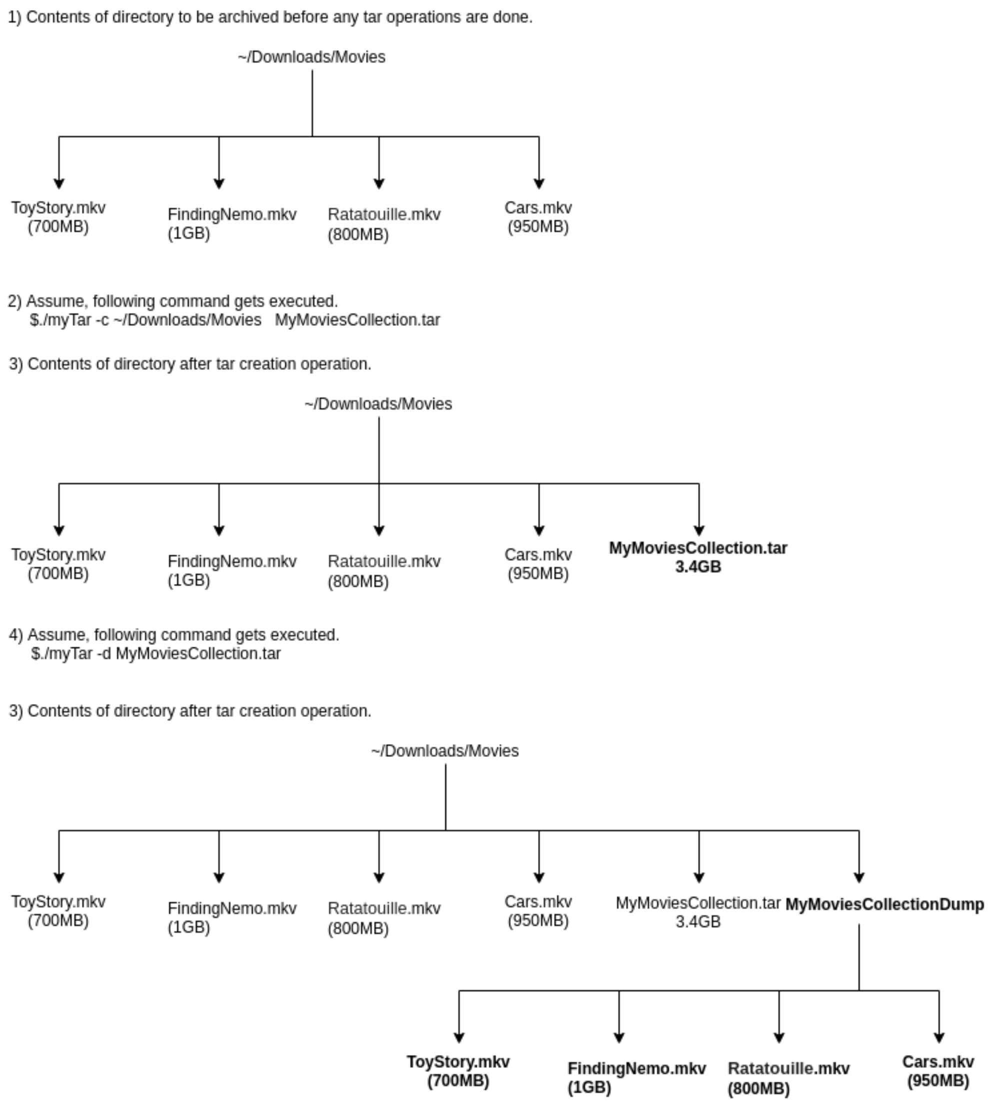
- For each file, it should store the contents of that file in output tar file. You can assume that output tar file doesn't already exists and you have to create it with permissions 0644 and fill it with data.
- You should store the contents of each file in tar file in such a manner that other operations i.e. extraction of files from tar files, listing of contents of tar file can be done correctly.

Assumptions

- You can assume that there is enough storage space present for archive file to get created.
- You can assume that during the creation of tar file, no new file will be added to or deleted from the directory being archived.

Error Handling

- In case of any error, print "Failed to complete creation operation"



1) Contents of directory to be archived before any tar operations are done.
~/Downloads/Movies
ToyStory.mkv (700MB)
FindingNemo.mkv (1GB)
Ratatouille.mkv (800MB)
Cars.mkv (950MB)
2) Assume, following command gets executed.
$./myTar -c ~/Downloads/Movies MyMoviesCollection.tar
3) Contents of directory after tar creation operation.
~/Downloads/Movies
ToyStory.mkv (700MB)
FindingNemo.mkv (1GB)
Ratatouille.mkv (800MB)
Cars.mkv (950MB)
MyMoviesCollection.tar 3.4GB
4) Assume, following command gets executed.
$./myTar -d MyMoviesCollection.tar
3) Contents of directory after tar creation operation.
~/Downloads/Movies
ToyStory.mkv (700MB)
FindingNemo.mkv (1GB)
Ratatouille.mkv (800MB)
Cars.mkv (950MB)
MyMoviesCollection.tar 3.4GB
MyMoviesCollectionDump
ToyStory.mkv (700MB)
FindingNemo.mkv (1GB)
Ratatouille.mkv (800MB)
Cars.mkv (950MB)

Figure 1: Example of tar file creation and extraction operations

2.1.2 Extraction of all archived files from a tar file

Command line arguments

```
    $./myTar -d <tar filepath>
Eg: $./myTar -d ~/Downloads/Movies/MyMoviesCollection.tar
```

Description of command line arguments

- myTar: simple tar utility that you have to implement
- -d: This argument tells your tar utility that extraction of all files operation needs to be done.
- tar filepath: Path of tar file on which extraction operation needs to be done.

Description of functionality

- Your tar utility should create a new directory named "tar filename(without .tar) + Dump" in the directory where tar file is present. Example: In figure 1, MyMoviesCollectionDump gets created in ~/Downloads/Movies directory. You can assume that no such directory already exists. This directory should be created with mode 0777.
- For each file, that was archived in the tar file, create a new file in this newly created directory. Name of this file should be same as its original name when it was archived and likewise its contents should also be restored to be same as the original contents. Permissions of each extracted file should be set as 0644.

Assumptions

- You can assume that there is enough storage space present for the contents of archive file to get extracted.

Error Handling

- In case of any error, print "Failed to complete extraction operation"

2.2 Listing the contents of tar file [20 Marks.]

Command line arguments

```
    $./myTar -l <tar filepath>
Eg: $./myTar -l ~/Downloads/Movies/MyMoviesCollection.tar
```

Description of command line arguments

- myTar: simple tar utility that you have to implement
- -l: This argument (it is character 'ell', not digit one) tells your tar utility that listing of the contents of tar file needs to be done.
- tar filepath : path to tar file whose contents need to be listed.

Description of functionality

- Create a file "tarStructure" with permissions 0644 in directory where tar file is present. You can assume that this file doesn't already exist.
- Write information about tar file such as size of tar file and number of files archived in tar file into "tarStructure" file.
- Scan tar file and write filename and the size of each archived file to "tarStructure" file.
- Above mentioned information should be written to "tarStructure" file in following format:

```
        size of tar file in bytes<newline>
        Number of files archived in tar<newline>
        filename<space>file size in bytes<newline>
        filename<space>file size in bytes<newline>
            .
            .
            .
    Eg: Output of following command:
            $./myTar -l MyMoviesCollection.tar
        will be:
            3643801600
            4
            ToyStory.mkv 734003200
            FindingNemo.mkv 1073741824
            Ratatouille.mkv 838860800
            Cars.mkv 996147200
```

Error Handling

- In case of any error, print "Failed to complete list operation"

2.3 System calls and library functions

You **must only use** the below mentioned APIs to implement this question.

```
- open          - close
- read          - write
- opendir       - closedir
- readdir       - mkdir
- chdir         - stat,fstat,lstat
- lseek         - exit
- malloc        - free
- memset        - printf
- str* family   - sprintf
```

2.4 Testing

To check whether your implementation of creation and extraction of tar file is working correctly, do make in Part2 directory and then run Part2/run_test.sh script. It creates a tar file with files present in Part2/test directory and then extracts them. It prints following output if your utility is implemented correctly:

```
File 1 extracted correctly
File 2 extracted correctly
File 3 extracted correctly
File 4 extracted correctly
```

3 Submission

- Make sure that your implementation doesn't print unnecessary data. Your output should match exactly with the expected output specified in each question.
- You have to submit zip file named your_roll_number.zip Eg: 1211405.zip containing **only** the following files in specified folder format:
  - YourRollno/Part1/square.c, YourRollno/Part1/double.c, YourRollno/Part1/root.c
  - YourRollno/Part2/myTar.c.