CS 451 / 551 Homework 2

Due: Mar 13 **Max Points:** 100
No Late Submissions

To be turned in on paper in class.

Please remember to justify all answers.

Important Reminder As per the course Academic Honesty Statement, cheating of any kind will minimally result in receiving an F letter grade for the entire course.

1. You have built a library based on your solution to prj2 (../../projects/prj2/prj2.html>) and your library has many users who access your library implementation using the interface defined in `matrix_mul`.h. Your users are happy with your library except for its error reporting; they find that the single `int` code returned via the `err` pointer does not provide sufficient information. They would like to get back a string error message which provides a detailed **context-sensitive** explanation for the error.

   You have also fixed a bug in your library. You create a new version of your library which contains the bug fix and also provides access to string error messages. What changes would you make to the library interface and implementation to meet the following constraints:

   - Users who are not interested in string errors but are interested in your bug fix should be able get access to your bug fix by simply relinking their client code with the new version of your library **without needing to change their source code**.
   - Users who are interested in string errors will make changes to their source code and relink with the new version of your library.
   - Some users have client code which have multiple `MatrixMul` objects live at the same time.

   This question deals only with the interface changes you need to make to satisfy the above constraints. It does not deal with the implementation of multiple worker processes, pipes, etc. *10-points*

2. A parent process creates `N` **interchangeable** resources which are to be shared among the parent and its descendent processes. A resource is referenced by its index `0, ..., (N-1)`. When a process needs a resource, it needs **exclusive** access to the resource. How would you design a resource manager for these `N` resources using pipes as your only IPC mechanism. You may assume that `N` is small (around 4). Clearly document any other assumptions in your design. *10-points*

3. Assuming that function and data pointers have the same representation, what would you expect the following function to return on a typical Unix system.

```
char c;

int f() {
  void *fP = &f;
  void *cP = &c;
  return ((cP < (void *)&cP)<<1) + (fP < (void *)&fP);
}
```

Please remember to justify your answer. *10-points*

4. Multiple processes need to send messages to a single process via a single shared pipe. It is possible that the size of a single message exceeds `PIPE_BUF` bytes. Design a protocol to be followed by the reading process and writing processes so that messages can be sent correctly even though the writing processes may be writing concurrently to the pipe. *10-points*

5. When you inherit a legacy application which does user authentication using passwords, you are horrified to find that user passwords are hashed using the insecure MD5 algorithm and stored in a relational database. You explain the seriousness of the problem to your manager. You suggest a fix which requires each user to change their password so that their new password can be hashed using a secure Blowfish hash. Your manager rejects your fix as it is too obtrusive for users; in fact, she says that she will reject any solution which involves user involvement.

   Describe how you would change the application so that all user passwords are protected using Blowfish without requiring users to change their passwords. Note that you cannot get access to the users' plaintext passwords. *10-points*

6. Assume that a process with real, effective and saved set-user ID 1000 performs the following sequence of steps:

   a) Successfully `exec`'s a non-setuid program.

   b) Successfully `exec`'s a setuid program owned by UID 2000.

   c) Calls `setuid(2000).`

   d) Calls `seteuid(1000).`

   e) Calls `setreuid(1000, 2000).`

   f) Calls `setreuid(1000, 1000).`

   Give the values of the real, effective and saved set-user-ID after each of the above steps. Please justify your answers. *10-points*

7. User `u1` belongs to primary group `g1` and supplementary group `g2`, and user `u2` belongs only to primary group `g2`. Neither `u1` nor `u2` are super-users, nor do they belong to any supplementary groups other than those explicitly listed above. Given the following `ls -l` listing:

```
-rwxr-xr-x    1 u2       g2          14096 Sep  8 22:11 exec1
-rwsr--r-x    1 u1       g1          44096 Sep 25 11:16 exec2
-rw----rw-    1 u2       g2           4012 Sep 23 03:10 data1
-rw-r--r--    1 u1       g1           8222 Sep  8 14:23 data2
```

fill in the following matrix with a `Y`, `N` or − depending on whether the execution specified by the row can (`Y`) or cannot (`N`) make the access specified by the column (R denotes *read*, W denotes *write*), or such access does not make sense (−).

	data1 R	**data1 W**	**data2 R**	**data2 W**
`u1` runs `exec1`				
`u2` runs `exec1`				
`u1` runs `exec2`				
`u2` runs `exec2`				

Please justify your answers. *10-points*

8. How would you set up `N` processes with a ring communication structure with all IPC handled using anonymous pipes. Specifically, if the processes are `P[0]`, `P[1]`, `P[N - 1]`, then `P[i]` should be able to send data to `P[(i + 1)%N]`. Your answer should detail exactly how you would create pipes (including closing of unneeded pipe ends). You can make your answer concrete by detailing the exact steps needed for `N == 4`. *10-points*

9. The text explains that all the other `exec()` functions are *layered on top of* `execve()` (pg. 567); i.e. all the other `exec()` functions are implemented using it as a primitive. Of the 6 `exec()` functions, is `execve()` the only choice as a primitive or could another `exec()` function be chosen as a primitive for implementing the rest? *10-points*

10. Discuss the validity of the following statements: *10-points*

    a) It is a good idea to close pipe file descriptors when they are no longer needed since a process is limited to a certain maximum number of open file descriptors.

    b) If a user with a particular user id creates a file and the permissions/ownership on that file are never changed, then that user can also remove the file.

    c) The user and group ownership of a new file is always set to the effective uid and gid of the process which created it.

    d) If a process has execute access to a directory, then it can search that directory.

    e) Once a process `exec`'s a setuid program it will always run with an effective uid equal to that of the owner of the program.