Ronak Nathani
CS 543: Operating Systems | Homework 3

Ans1.

Yes, a command line shell can be written using threads, as threads are a component of a process. A process with just one thread is a lightweight process whereas a process can also have multiple threads. A few things change with this implementation, as the thread of a process uses the same address space, stacks and memory as the parent process whereas, two different processes have different stacks, address space and memory. Moreover, thread ids will have to be considered in this case instead of pid.

Advantages-

1. Threads are lightweight in comparison to processes as they require far less resources than processes. It is comparatively easier to create threads.

2. Threads don't require a separate address space and hence use less memory.

3. CPU cycles could be used more efficiently by creating multiple threads on a multi core system. Now, even a process can also create multiple children and have them execute on a multi core system simultaneously but doing so with threads is comparatively easier because of low resources & low memory required by threads. Threads serve better for parallelism.

4. The overhead of context switching in threads is very less compared to the overhead of context switching in processes. A process can have various children processes or a process can also have multiple threads instead. In case of a process having multiple threads, the context switching is a lot faster and results in a faster turnaround.

5. When a child process is created using a fork(), the parent process waits for it to terminate unless otherwise it is specified to run the parent process concurrent to the child. In case of threads, a parent process can have multiple threads running concurrently not needing any explicit instructions.

6. In a thread based command line shell, it could possible have command auto-completion running alongside another thread accepting user inputs, yet another thread running in the background executing a command, a thread creating the log of the session, etc.

7. Any interprocess communication won't be required as threads share the same address space and memory.

Disadvantages-

1. There would be times when one thread would need to be independent requiring its own separate address/memory. This won't be possible with threads.
2. Since, threads share the same memory and address space they are very much prone to race conditions, starvation and deadlocks. A deadlock can also occur if multiple processes are allowed to run and they tend to share resources. But in case of threads sharing address space, memory and stack is by default. In that case, the chances of race conditions and deadlocks is higher in concurrent threads than concurrent processes.

3. In order to deal with race conditions and deadlocks we would need to implement semaphores, mutual exclusion etc. Avoiding these problems is not a trivial task and leads to complex coding requirements.

4. These coding requirements make the code maintenance much more difficult than in the case of processes.

5. Security is a major issue due to extensive sharing in threads.

6. If the kernel is single threaded, then the system call of one thread will block the whole process and the CPU would be idle.

7. Many libraries are not thread safe.

8. Inadvertent changes in global variables can be disastrous when dealing with threads as everything is shared.

9. Memory crash in one thread will kill other threads sharing same memory. This is not the case with processes.

Ans 2.

In the current command shell implementation, the commands entered by the user are first checked against the internal shell keywords and if they match any one of the keywords then they are not checked against the system programs. Hence, this avoids any collisions. This is the approach I used in this program to first check the user commands against the internal keywords and if there is no match, only then move forward to check them against the system programs.

There could be other approaches too, like-

1. Namespace- To assign names to separate groups so that the names differ in absolute terms.

2. Renaming – If there is a collision then changing the name of the internal shell keyword using a convention.

3. Prefixing- Using unique characters before the names of the variables as a convention so that the names differ and any collisions are avoided to happen by chance.

In this command shell implementation, as the user tries to create an alias by entering command of the form 'alias aliasName “commandName”' the program checks the first argument, 'alias', against the internal shell keywords and finds a match. As soon as there is a match, the program avoids any further checks against the system programs or other internal shell keywords. Once the keyword is matched against, the other arguments are parsed to create an alias at the user level program, i.e. the command line shell interpreter.

Ans 3.

To be able to run a shell under different usercode, we would need to add a feature to our shell which allows us to access the /etc/ passwd file. In that case a particular command could be created to allow user to do this, for instance ‘chuser’. When this command is entered, the shell would prompt for a login and a password. Once login and password are entered, we can look up the login name in /etc/passwd folder and verify the password. Also, we can have the shell display the current usercode.

We would also have the permissions associated with every usercode. Hence, allowing only authorized users to access files they are permitted to access. The program would give an error message if the current usercode doesn't have the permission to access a certain command or a file.

Moreover, we could have a feature where the current usercode can act as another usercode just for a particular command or to run a particular script. It wouldn't change the entire usercode but let the current usercode act as the requested usercode to invoke a particular request. In order to be able to act as another usercode, the current usercode will be prompted for the password of the usercode it is requesting to act as. Only if the password and login are verified, the current usercode would be permitted to act as another usercode.

Also, this feature would allow the current usercode to attempt the password of the new usercode just for a certain number of times, usually 3, until the correct password is entered. Otherwise, the new usercode request is to be re-entered followed by password prompts.

The second interface suggested could be just like running the sudo command. In that case, the usercode to be requested can be entered at the command prompt followed by the request and the password.