Daemons

Presented by:
Shahzaib Irfan, 2021-CS-7
Afraz Butt, 2021-CS-12

Introduction

What is a Daemon?

A daemon (pronounced DEE-muhn) is a program that runs continuously as a background process and wakes up to handle periodic service requests, which often come from remote processes.

Usefulness

Daemons respond to network, hardware, or system requests to perform certain tasks, asynchronously.

Common Examples

- Wallpaper loading service in mobiles and systems
- Apache2: a web server daemon on Ubuntu-powered systems

Background Process vs Daemons

- Background Process usually occurs in the context of shell for job control.
- Daemons have no controlling terminal and are made to be child of init process.

Daemon Lifecycle

- Start at boot time
- Ends when system shuts down

Daemonizing a process

Daemonization in C

- Fork a child process from the parent process using the fork() system call.
- In the child process, create a new session using the setsid() system call to detach the child process from its controlling terminal.

.

Daemonization in C *(contd.)*

- Change the working directory of the child process to the root directory using the chdir() system call to avoid any issues with unmounting file systems.
- Close all open file descriptors inherited from the parent process using the close() system call, except for the standard input, output, and error file descriptors (stdin, stdout, and stderr).

Daemonization in C (contd.)

- Redirect the standard input, output, and error file descriptors to /dev/null using the dup2() system call to prevent any output from being sent to the terminal.
- Write the process ID (PID) of the child process to a file for future use.

Best Practices

Logging

- Debugging problems is important in daemons because they have no terminal attached with them.
- Using syslog or log4c (libraries) to log events
- Logging events in *files* rather than consoles

Security

A daemon is a potential security risk because of its asynchronus state.

Possible solutions might include:

Security (contd.)

- Changing group and user permissions for daemon after it starts
- Change level to non-root user
- Dropping un-necessary priviliges for daemons such as file access, network access etc.

Conclusion

- Daemons are important part of rendering backend services.
- By using best practices as outlined, developers can make robust daemons providing essential services to consumers and admins alike.

Tutorial in accompanying typora file.

Thank you! Any questions?