Linux Daemon Processes

Agenda

- Introduction to Deamon Processes
- Daemon Process - Mandatory Requirements
- Daemon Process – Additional Requirements
- Advanced Daemon process – Optional Requirements
- Examples

Introduction to Daemon processes

- ➢ Used for programs which provide services.
- ➢ Used for programs which is expected to run forever (from boot to shutdown).
- ➢ Runs in infinite loop.
- ➢ Runs free from controlling terminal.
- ➢ Used for system services.
- ➢ Used for network services.

Daemon Processes – Mandatory Requirements

- Must run in background.
- Must be free from controlling terminal.
- Must belong to a separate session & separate process group.
- Must not be able to re-acquire a terminal.
- Must not be session leader and process group leader.
- Must close file descriptors 0 (std. input), 1 (std. output), 2 (std. error).

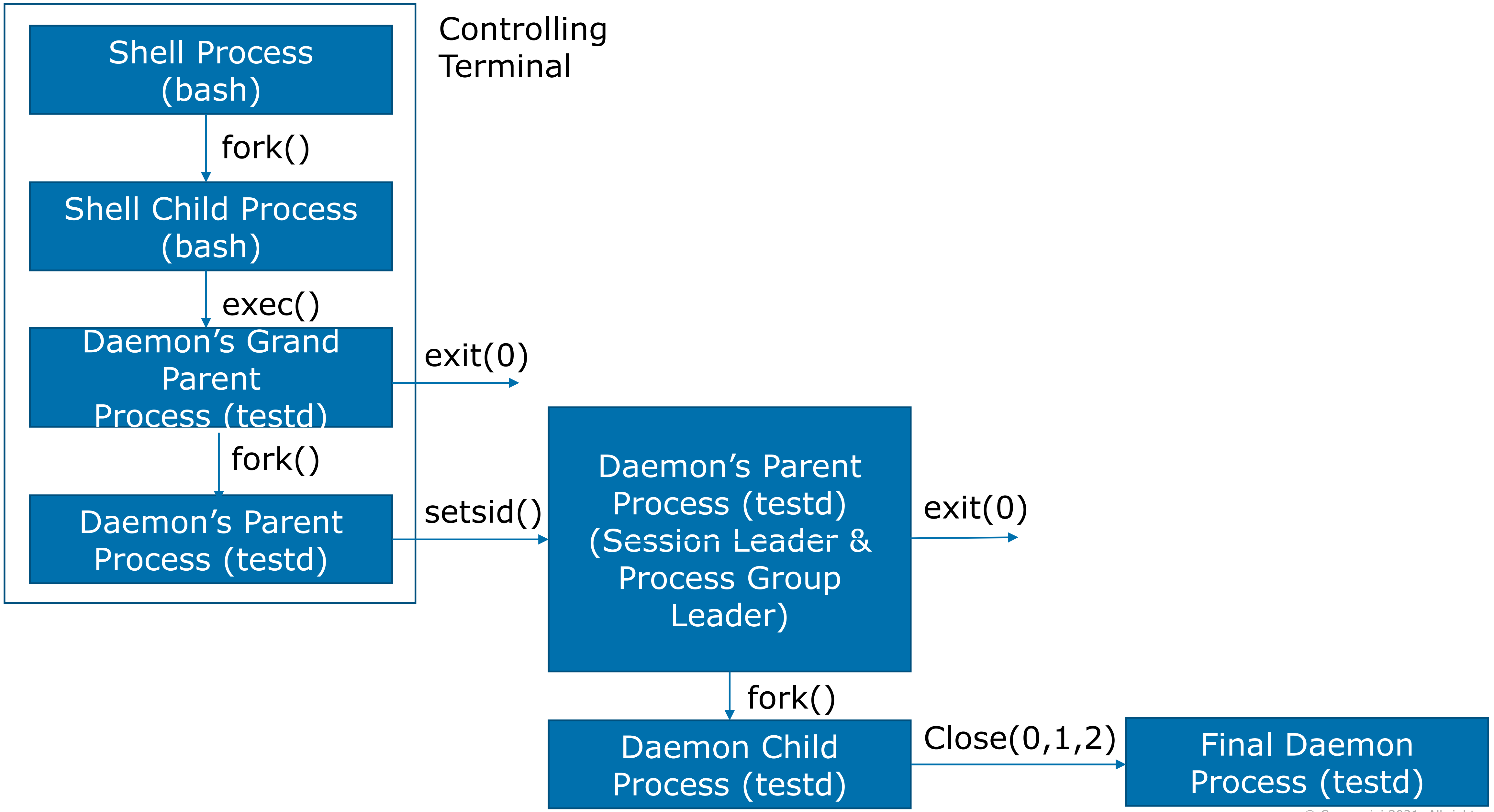
Daemon Processes – Additional Requirements

- **Robustness:**
  - ✓ Should change working directory to “/”.
- **Security:**
  - ✓ Should set umask to 0 or any appropriate value.
  - ✓ Should drop the privileges appropriately.
- **Efficient:**
  - ✓ Should close all inherited file descriptors.
  - ✓ Should remove all inherited custom signal handlers and replace with default signal handlers.
  - ✓ should remove all inherited environment variables?

Advanced Daemon Processes – Optional Requirements

- May log messages to syslog or a separate log file.
- May log pid into a separate file (usually <daemon name>.pid) to be used by start-up script (RC script) or systemd services.
- May provide a start/stop/restart script for systemd services.
- May implement SIGHUP signal handler to re-read the configuration file.
- May implement SIGTERM to stop or restart the process.
- May comply with systemd services.

Basic Daemon Process Creation


Controlling Terminal
Shell Process (bash)
fork()
Shell Child Process (bash)
exec()
Daemon's Grand Parent Process (testd)
exit(0)
fork()
Daemon's Parent Process (testd)
setsid()
Daemon's Parent Process (testd) (Session Leader & Process Group Leader)
exit(0)
fork()
Daemon Child Process (testd)
Close(0,1,2)
Final Daemon Process (testd)

Basic Daemon-1

```
#include <stdio.h>
#include <unistd.h>
main()
{
daemon(0,0); // BSD style daemon
for(;;)
{
sleep(10);
}
// Never reached.
//exit(0);
}
```

Basic Daemon-2 (page-1)

```
#include <stdio.h>
#include <unistd.h>
void daemonization()
{
int pid;
pid = fork();
if (pid < 0){
        printf("Failed to create child process\n");
        exit (-1);
}
else if (pid > 0)
{
// Grand Parent process of the Daemon process must exit.
        exit(0);
}
// Free the child (parent process of Daemon process) from controlling terminal and
// make it the process group leader and session leader.
if(setsid() < 0){
        printf("Failed to free the process from controlling terminal.\n");
        exit (-1);
}
```

Basic Daemon-2 (page-2)

```
pid = fork();
if (pid < 0){
                    printf("Failed to create child process\n");
                    exit (-1);
}
else if (pid > 0)
{
//Parent process of the Daemon process must exit.
                    exit(0);
}
// Code at this point is the Grand child process (the daemon process).
// Close standard input, standard output and standard error devices.
close(0);
close(1)
close2);
}
main()
{
daemonization();
// TODO: Additional operations to make it robust, secure & efficient daemon process.
for(;;)
{
// Perform some periodic task.
sleep(10);
}
// Never reached.
//exit(0);
}
```