

CS810 - Advanced Programming in the UNIX Environment

—

Dæmon processes

Department of Computer Science
Stevens Institute of Technology
Jan Schaumann

`jschauma@cs.stevens.edu`

`http://www.cs.stevens.edu/~jschauma/810-APUE/`

Client-Server Model

- two categories of servers
 1. iterative
 2. concurrent

Client-Server Model

- two categories of servers
 1. iterative
 - 1.1. wait for client request to arrive
 - 1.2. process the client request
 - 1.3. send the response back to the client
 - 1.4. go back to 1.1
 2. concurrent

Client-Server Model

- two categories of servers
 1. iterative
 - 1.1. wait for client request to arrive
 - 1.2. process the client request
 - 1.3. send the response back to the client
 - 1.4. go back to 1.1
 2. concurrent
 - 2.1. wait for client request to arrive
 - 2.2. start a new server to handle this client's request
 - 2.3. go back to 2.1

Dæmon characteristics

Commonly, dæmon processes are created to offer a specific service.

Dæmon processes usually

- live for a long time
- are started at boot time
- terminate only during shutdown
- have no controlling terminal



BSD Daemon Copyright 1988 by Marshall Kirk McKusick
All Rights Reserved.

Dæmon characteristics

The previously listed characteristics have certain implications:

- do one thing, and one thing only
- no (or only limited) user-interaction possible
- consider current working directory
- how to create (debugging) output



BSD Daemon Copyright 1988 by Marshall Kirk McKusick
All Rights Reserved.

Writing a dæmon

- fork off the parent process
- change file mode mask (umask)
- create a unique Session ID (SID)
- change the current working directory to a safe place
- close (or redirect) standard file descriptors
- open any logs for writing
- enter actual dæmon code



BSD Daemon Copyright 1988 by Marshall Kirk McKusick
All Rights Reserved.

Dæmon conventions

- prevent against multiple instances via a *lockfile*
- allow for easy determination of PID via a *pidfile*
- configuration file convention */etc/name.conf*
- include a system initialization script (for */etc/rc.d/* or */etc/init.d/*)
- re-read configuration file upon SIGHUP



BSD Daemon Copyright 1988 by Marshall Kirk McKusick
All Rights Reserved.

A central logging facility

There are three ways to generate log messages:

- via the kernel routine `log(9)`
- via the userland routine `syslog(3)`
- via UDP messages to port 514

syslog(3)

```
#include <syslog.h>

void openlog(const char *ident, int logopt, int facility);
void syslog(int priority, const char *message, ...);
```

openlog(3) allows us to set specific options when logging:

- prepend *ident* to each message
- specify logging options (LOG_CONS | LOG_NDELAY | LOG_PERRO | LOG_PID)
- specify a *facility* (such as LOG_DAEMON, LOG_MAIL etc.)

syslog(3) writes a message to the system message logger, tagged with *priority*. A *priority* is a combination of a *facility* (as above) and a *level* (such as LOG_DEBUG, LOG_WARNING or LOG_EMERG).