CpSc 360: Distributed and Network Programming

Daemons and Inetd

James Wang

Textbook chapter 13

Daemons

- **A daemon is a process that:**
  - runs in the background
  - not associated with any terminal
    - output doesn't end up in another session.
    - terminal generated signals (^C) aren't received.
- **Unix systems typically have many daemon processes.**
- **Most servers run as a daemon process.**

Common Daemons

- **Web server (httpd)**
- **Mail server (sendmail)**
- **SuperServer (inetd)**
- **System logging (syslogd)**
- **Print server (lpd)**
- **router process (routed, gated)**

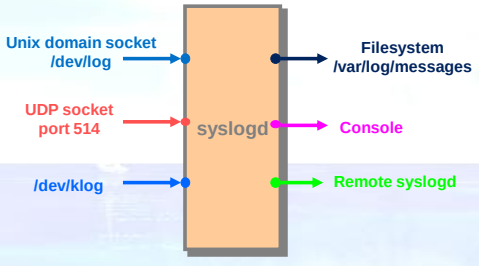
Daemon Output

- **No terminal - must use something else:**
  - file system
  - central logging facility
- **Syslog is often used - provides central repository for system logging.**

Syslog service

- **`syslogd` daemon provides system logging services to "clients".**
- **Simple API for "clients"**
  - A library provided by O.S.
- **A system administrator can control logging functions by specifying:**
  - where messages should go
  - what kinds of messages are important
  - what can be ignored

`syslogd`



Syslog messages

- **Think of syslog as a server that accepts messages.**
- **Each message includes a number of fields, including:**
  - a *level* indicating the importance (8 levels)
    - **`LOG_EMERG`** highest priority
    - **`LOG_DEBUG`** lowest priority
  - a *facility* that indicates the type of process that sent the message:
    - **`LOG_MAIL, LOG_AUTH, LOG_USER, LOG_KERN, LOG_LPR,...`**
  - A text *string*.
- **Message: (*level*,*facility*,*string*)**

/etc/syslog.conf

- **Syslogd reads a configuration file that specifies how various messages should be handled (where they should go).**
- **The sysadmin controls all logged messages by editing this file.**
- **Examples:**
  - Sysadmin could set LOG_EMERG messages to be sent to the console
  - low priority messages from lpr could be thrown away.
  - Medium priority message from the mail server could be saved in a file.

Sending a message to syslogd

- **Standard programming interface provided by `syslog()` function:**

```
#include <syslog.h>
void syslog( int priority,
             const char *message,
             . . . );
```

- **Works like `printf()`**

Syslog client/server

- **Clients send messages to local syslogd through a unix domain (datagram) socket.**
- **All the details are handled by `syslog()`**
- **`syslogd` sends/receives messages to/from other hosts using UDP.**

How to create daemons?

- **To force a process to run in the background, just fork() and have the parent exit.**
- **There are a number of ways to disassociate a process from any controlling terminal.**
  - Call **`setsid()`** and then **`fork()`** again.

Example

```
#include <sys/types.h>
#include <sys/stat.h>
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <errno.h>
#include <unistd.h>
#include <syslog.h>
#include <string.h>

int main(void) {

    /* Our process ID and Session ID */
    pid_t pid, sid;

    /* Fork off the parent process */
    pid = fork();
    if (pid < 0) {
        exit(EXIT_FAILURE);
    }
    /* If we got a good PID, then
       we can exit the parent process. */
    if (pid > 0) {
        exit(EXIT_SUCCESS);
    }

    /* Change the file mode mask */
    umask(0);

    /* Open any logs here */
```

```
    /* Create a new SID for the child process */
    sid = setsid();
    if (sid < 0) {
        /* Log the failure */
        exit(EXIT_FAILURE);
    }



    /* Change the current working directory */
    if ((chdir("/")) < 0) {
        /* Log the failure */
        exit(EXIT_FAILURE);
    }

    /* Close out the standard file descriptors */
    close(STDIN_FILENO);
    close(STDOUT_FILENO);
    close(STDERR_FILENO);

    /* Daemon-specific initialization goes here */

    /* The Big Loop */
    while (1) {
      /* Do some task here ... */

      sleep(30); /* wait 30 seconds */
    }
   exit(EXIT_SUCCESS);
}
```

Daemon initialization

- Daemons should close all unnecessary descriptors
  - often including `stdin, stdout, stderr`.
- Get set up for using syslog
  - Call `openlog()`
- Often change working directory.

13

Too many daemons?

- There can be many servers running as daemons - and idle most of the time.
- Much of the startup code is the same for these servers.
- Most of the servers are asleep most of the time, but use up space in the process table.
- Most Unix systems provide a "SuperServer" that solves the problem:
  - executes the startup code required by a bunch of servers.
  - Waits for incoming requests destined for the same bunch of servers.
  - When a request arrives - starts up the right server and *gives it the request*.

14

`inetd`

- The SuperServer is named `inetd`. This single daemon creates multiple sockets and waits for (multiple) incoming requests.
- `inetd` typically uses `select` to watch multiple sockets for input.
- When a request arrives, `inetd` will fork and the child process handles the client.

15

`inetd` children

- The child process closes all unnecessary sockets.
- The child `dup`'s the client socket to descriptors 0,1 and 2 (`stdin, stdout, stderr`).
- The child `exec`'s the real server program, which handles the request and exits.

16

`inetd` based servers

- Servers that are started by `inetd` assume that the socket holding the request is already established (descriptors 0,1 or 2).
- TCP servers started by `inetd` don't call `accept`, so they must call `getpeername` if they need to know the address of the client.

17

`/etc/inetd.conf`

- `inetd` reads a configuration file that lists all the services it should handle.
- `inetd` creates a socket for each listed service, and adds the socket to a `fd_set` given to `select()`.
- **Example:**

```
......

echo     stream  tcp  nowait  root  internal
echo     dgram   udp         wait    root  internal
chargen  stream  tcp         nowait  root  internal
chargen  dgram   udp         wait    root  internal
ftp      stream  tcp         nowait  root  /usr/sbin/ftpd ftpd -l
telnet   stream  tcp  nowait  root  /usr/sbin/telnetd telnetd
finger   stream  tcp         nowait  root  /usr/sbin/fingerd fingerd

......
```

18

`inetd` service specification

- **For each service, `inetd` needs to know:**
  - the port number and transport protocol
  - wait/nowait flag.
  - login name the process should run as.
  - pathname of real server program.
  - command line arguments to server program.

19

wait/nowait

- **Specifying `wait` means that `inetd` should not look for new clients for the service until the child (the real server) has terminated.**
- **TCP servers usually specify `nowait` - this means inetd can start multiple copies of the TCP server program - providing concurrency!**
- **Most UDP services run with `inetd` told to wait until the child server has died.**
  - **What would happen if:**
    - **`inetd`** did not wait for a UDP server to die?
    - **`inetd`** gets a time slice before the real server reads the request datagram?

20

UDP Servers that wait/nowait

- **Some UDP servers hang out for a while, handling multiple clients before exiting.**
- **`inetd` was told to wait – so it ignores the socket until the UDP server exits.**

21

Super inetd

- **Some versions of inetd have server code to handle simple services such as**

  echo server, daytime server, chargen…
- **Servers that are expected to deal with frequent requests are typically *not* run from inetd: mail, web, NFS.**
- **Many servers are written so that a command line option can be used to run the server from `inetd`.**
- **Some versions of Unix provide a service very similar to `inetd` called `xinetd`.**
  - configuration scheme is different
  - basic idea (functionality) is the same…

22