

Handling Children for the Concurrent Server

Unix Network Programming
Ch # 5

Today's Agenda

- Lab 7 will be due next Tuesday April 15
- Stevens Ch #5 & 6 Client Server examples
-

Reminders: Talks this Week

1) Monday April 7th, 1:00 pm Science 127

Dr. Nancy Amato

2nd Annual Computational Sciences Invited Lecture Series

From Intelligent CAD to Computer Animation to Protein Folding

2) Tuesday April 8th , 3:30 pm Jour 129

Dr. Ray Hamid

Candidate, Head Department Computer Science & Information Systems

Intelligent Systems

3) Thursday April 10th, 11:00 am Jour 234 (Media Room)

Dr. James Jones,

Candidate, Faculty Position Department Computer Science & Information Systems

Data Structures

4) Thursday April 10th, 3:30 pm Jour 129

Dr. Jack Yang

Candidate, Faculty Position Department Computer Science & Information Systems

Bioinformatics

Simple Echo Server

- valid, yet simple example of a network application
- all the basic steps required to implement any client/server are illustrated
- to expand example for a real application, you would change what the server does with the input it receives from its clients.
- Fig 5.1

Normal Termination

- `tcpserv01`
- `netstat -a | grep 9877`
- `tcpcli01 127.0.0.1`
- `netstat -a | grep 9877`
- `ps -t pts/6 -o pid,ppid,tty,stat,args,wchan`
- The `STAT` of the child is now `Z` (for zombie)
 - Need to clean up our zombie processes, and doing this requires dealing with Unix signals.

POSIX Signal Handling

- Signal is a notification to a process that an event has occurred.
- Software interrupts
- Signals occur *asynchronously*
- Signals are sent by:
 - one process to another process (or to itself)
 - the kernel to a process
- SIGCHLD signal is sent by the kernel whenever a process terminates, to the parent of the terminating process

POSIX Signal Handling

- Every signal has a disposition, which is also called the action associated with the signal.
 - Can provide a function that is called whenever a specific signal occurs (except SIGKILL and SIGSTOP).
 - Can ignore a signal by setting disposition to SIG_IGN
 - Can set to default disposition by using SIG_DFL
- signal() function, POSIX standard is now sigaction() function.

Handling SIGCHLD Signals

- Recall our concurrent server, child processes became zombies
- Purpose of zombie state is to maintain information about the child for the parent to fetch at some later time.
- Don't want to leave zombies around, they take up space in the kernel process tables
- Whenever we fork a child, we must wait for them to get their final statuses
- To do this we establish a signal handler to catch SIGCHLD, and within the handler we call wait

Handling SIGCHLD Signals

- tcpsrv02

Handling Interrupted System Calls

- Problem: A signal can occur when we are blocked in a system call, such as `accept`, or `read`
 - “slow system call”
- Problem: some OS will automatically restart the system call, for others the system call returns with an error of `EINTR` (interrupted system call)
- For portability, when a process is blocked in a slow system call, and the process catches a signal and the signal handler returns, then the system call can return an error of `EINTR`
 - The basic rule for portability is that we should be aware of and handle this situation.

wait and waitpid

- Both return 2 values
 - the return value of the function is the process id of the child
 - termination status of child is returned in statloc

Bug with original Signal Handler

- Signals are not queued in Unix
- need to use `waitpid` function in a loop in order to handle correctly, if multiple `SIGCHLD` occur about the same time (before signal handler).