Topics: Spotlight on Servers

Approach: Two major Examples: timeserver, mini web server

Today’s System Calls: socket(), bind(), listen(), accept(), connect()

Outline

Once one used files and programs on one computer.
Now files and programs can be anywhere. How does it work?

Overview of Client-Server Programming

Abstracting the main functional units into separate functions.
make_server_socket(portnum)
connect_to_server(hostname, portnum)
process_request(fd) - talk_to_server(fd)

Server Design

The Time Service

1. purpose
2. big picture
3. the server: process_request
   takes a call
   computes the time, tells the caller, hangs up
4. the client: talk_to_server
   makes a call
   copies reply to stdout

An Alternate Time Service

5. the server: process request
   takes a call
   forks - child redirects stdout, execs date
   parent waits (?)

Pros and Cons of Using New Processes

A Mini Web Server

What is a web server? - A remote shell
   allows ls, cat, and exec remotely

Building a Web Server

The Main steps
The Protocol: explore it with telnet (see rfc1945)
The Main Loop: take request, send reply
Processing the Request
   error handling
   listing directories
   executing programs
   displaying files
A Web Server: The Big Picture and Code Outline

main
  setup
    → accept
    handle_call

handle_call
  fork
  Par
  Child
  read_request
  process_rq
  close
  exit

process_rq
  parse args
  modify argument
  cmd != "GET"
  cannot_do
  arg not exist?
  404
  arg is a directory?
  do_ls(arg)
  arg ends in "cgi"
  do_exec(arg)
  default
  do_cat(arg)

do_ls
  exec ls -l arg

do_exec
  exec arg

do_cat
  p = fopen arg
  while( getc(p) )
  putc(sock)
  fclose(p)
  fclose(sock)