


Final Exam Review


1. Final Review

1.1 CSCI 330

CSCI 330 UNIX and Network Programming



Final Exam Review



1.2 Topics covered before Midterm

Topics covered before Midterm

- Introduction
- File system
- Editors
- Net Utilities
- Permissions
- Shell
- Shell scripts
- sed stream editor
- awk report generator

1.3 History of UNIX

History of UNIX

- Invented by Ken Thompson at Bell Labs in 1969
 - first version written in assembly language
 - single user system, no network capability
- Thompson, Dennis Ritchie, Brian Kernighan
 - created C programming language, rewrote Unix in C
- Unix evolution:
 - Bell Labs, USL, Novell, SCO
 - BSD, FreeBSD, Mach, OS X
 - AIX, Ultrix, Irix, Solaris, ...
 - Linux: Linus Torvalds
- Newest:
 - Linux on portables:
Android

1.4 RTFM: The man Command

RTFM: The man Command

- show pages from system manual

Syntax: `man [options] [-S section] command-name`

```
% man date
% man -k date
% man crontab
% man -S 5 crontab
```

Caveats:

- Some commands are aliases
- Some commands are part of shell

Section	Description
1	User commands
2	System calls
3	C library functions
4	Special system files
5	File formats
6	Games
7	Misc. features
8	System administration

1.5 New since Midterm

New since Midterm

- C++ programming: IO management
- Systems programming: process & pipe
- Networking concepts
- UDP programming
- TCP programming
- Shell job control
- System administration

1.6 C Library Functions

C Library Functions

- C library is accessible to C++
- I/O operations: `#include <cstdio>`
 - functions: printf, scanf, putchar, getchar, ...
- Strings: `#include <cstring>`
 - functions: strlen, strcat, strcpy, strstr, memset, ...
- Standard general utilities: `#include <cstdlib>`
 - functions: atoi, rand, malloc, free, getenv, exit, system, ...
- Reference: <http://www.cplusplus.com/reference/clibrary>

1.7 System Call Categories

System Call Categories

- process control
 - create/terminate process, wait/signal event, allocate/free memory
- file management
 - create/delete file, open/close, read/write, get/set attributes
- device management
 - attach/request/release/detach device, read/write/position
- information management
 - get/set system data and attributes, e.g. time
- communication
 - create/delete connection, send/receive messages, remote devices

1.8 File Management

File Management

- | | |
|-------|-----------------------|
| open | open a file |
| read | read data from a file |
| write | write data to a file |
| close | close a file |
- also:

creat	make a new file
unlink	remove file
stat	get file information
chmod	change permissions
dup	duplicate file descriptor
 - all calls share file descriptor, i.e. number, to identify file

1.9 Process Management System Calls

Process Management System Calls

- fork
 - create a new process
- wait
 - wait for a process to terminate
- exec
 - execute a program
- pipe
 - establish communication channel
- dup
 - duplicate file descriptor

1.10 System Call: fork

System Call: fork

- creates new process that is duplicate of current process
- new process is almost the same as current process
 - difference: fork returns different values
- new process is child of current process
- old process is parent of new process
- after call to fork, both processes run concurrently

1.11 System Call: exec

System Call: exec

- family of functions that replace current process image with a new process image
 - actual system call: **execve**
 - library functions
 - **execl, execlp, execl**
 - **execv, execvp**
- arguments specify new executable to run and its arguments and environment

1.12 Together: fork and exec

Together: fork and exec

- UNIX does not have a system call to spawn a new additional process with a new executable
- instead:
 - fork to duplicate current process
 - exec to morph child process into new executable

1.13 UNIX Pipe

UNIX Pipe



- can create a software pipeline:
 - set of processes chained by their standard streams
- output of one process becomes input of second process

command line example:

```
ls | wc
```

implemented via **pipe** system call

1.14 System Call: pipe

System Call: pipe

```
int pipe(int pipefd[2])
```

- creates a channel to transport data
- has direction: one side to write, one side to read
 - available via 2 file descriptors `pipefd[2]`
 - read side `pipefd[0]`
 - write side `pipefd[1]`
- can be used synchronize producer and consumer of data

1.15 System Calls: pipe, fork, dup and exec

System Calls: pipe, fork, dup and exec

- Idea:
 - 2 processes communicate via pipe
 - each process exec's into new executable
- Example: `ls | wc`
 - parent: runs `ls`
 - child: runs `wc`

1.16 Networking: Layered protocols

Networking: Layered protocols

- complexities of communication organized into successive layers of protocols
 - lower-level layers: specific to medium
 - higher-level layers: specific to application
- standards achieve inter-operability

Open Systems Interconnection model (OSI reference model)

1.17 OSI reference model layers

OSI reference model layers



- provides services directly to user applications
- performs data transformations to provide common interface for user applications
- establishes, manages and ends user connection
- provides functions to guarantee reliable network link
- establishes, maintains and terminates network connections
- ensures the reliability of link
- controls transmission of the raw bit stream over the medium

1.18 Network Layer

Network Layer

- also called: Internet Protocol Layer
 - provides host to host transmission service, where hosts are not necessarily adjacent
- layer provides services:
 - addressing
 - hosts have global addresses: IPv4, IPv6
 - uses data link layer protocol to translate address: ARP
 - routing and forwarding
 - find path from host to host

1.19 Transport Layer

Transport Layer

- provides end-to-end communication services for applications
- byte format as abstraction on underlying system format
- raises reliability
- enables multiplexing:
 - provides multiple endpoints on a single node: port
 - refines connection address via port number

1.20 Transport layer ports

Transport layer ports

- 0 to 1023: well-known ports
 - 20 & 21: File Transfer Protocol (FTP)
 - 22: Secure Shell (SSH)
 - 23: Telnet remote login service
 - 25: Simple Mail Transfer Protocol (SMTP)
 - 53: Domain Name System (DNS) service
 - 80: Hypertext Transfer Protocol (HTTP) used in the World Wide Web
 - 110: Post Office Protocol (POP3)
 - 119: Network News Transfer Protocol (NNTP)
 - 143: Internet Message Access Protocol (IMAP)
 - 161: Simple Network Management Protocol (SNMP)
 - 443: HTTP Secure (HTTPS)
- 1024 to 49151: IANA registered ports
- 49152 to 65535: dynamic or private port

1.21 Domain Names

Domain Names

- hierarchical distributed naming system
- uses FQDN: fully qualified domain name
 - ex: faculty.cs.niu.edu
- DNS: domain name service
 - resolves query for FQDN into IP address
 - ex.: 131.156.145.186
- `getaddrinfo()` translates FQDN into IP address

1.22 Transport Layer

Transport Layer

- provides end-to-end communication services for applications
- provides multiple endpoints on a single node: port
- TCP: transmission control protocol
 - connection oriented, guaranteed delivery
 - stream oriented: basis for: http, ftp, smtp, ssh
- UDP: user datagram protocol
 - best effort
 - datagram oriented: basis for: dns, rtp

1.23 UDP programming

UDP programming

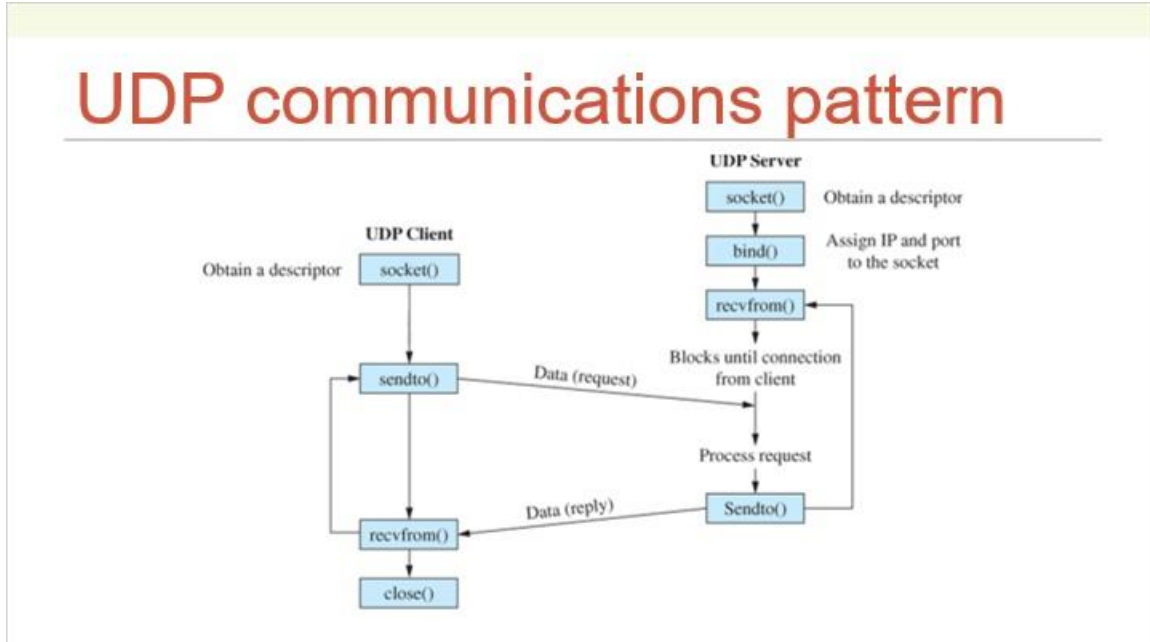
- common abstraction: socket
- first introduced in BSD Unix in 1981
- socket is end-point of communication link
 - identified as IP address + port number
- can receive data
- can send data

1.24 UDP Socket system calls

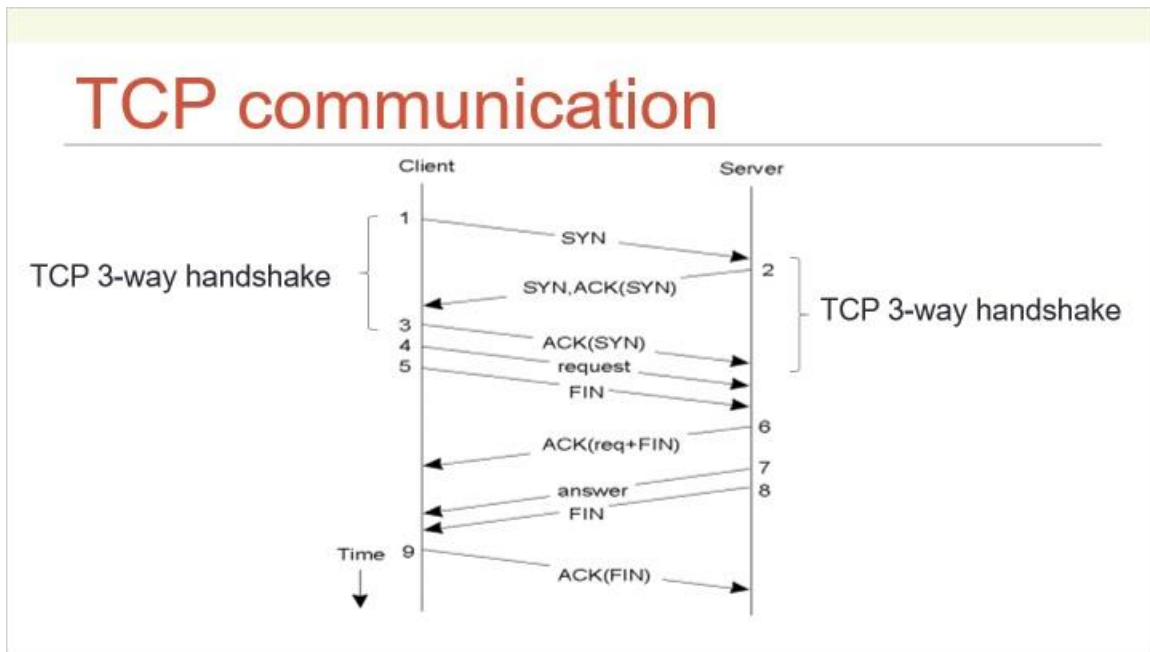
UDP Socket system calls

server	<table><tr><th>System call</th><th>Meaning</th></tr><tr><td>socket</td><td>Create a new communication endpoint</td></tr><tr><td>bind</td><td>Attach a local address to a socket</td></tr><tr><td>sendto</td><td>Send(write) some data over the connection</td></tr><tr><td>recvfrom</td><td>Receive(read) some data over the connection</td></tr><tr><td>close</td><td>Release the connection</td></tr></table>	System call	Meaning	socket	Create a new communication endpoint	bind	Attach a local address to a socket	sendto	Send(write) some data over the connection	recvfrom	Receive(read) some data over the connection	close	Release the connection	client
System call	Meaning													
socket	Create a new communication endpoint													
bind	Attach a local address to a socket													
sendto	Send(write) some data over the connection													
recvfrom	Receive(read) some data over the connection													
close	Release the connection													
↓		↓ optional ↓												

1.25 UDP communications pattern



1.26 TCP communication



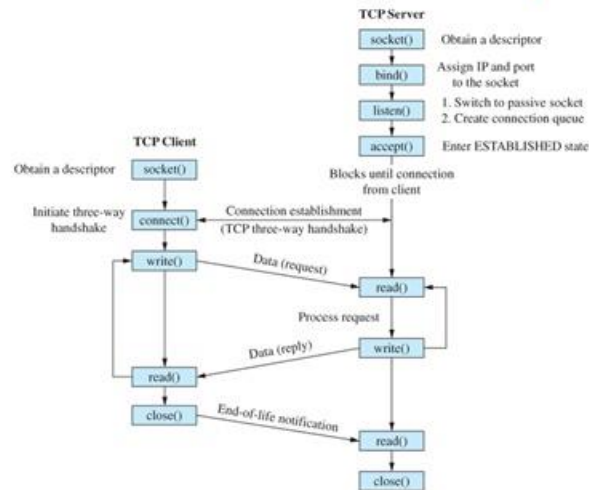
1.27 TCP Socket system calls

TCP Socket system calls

Primitive	Meaning
socket	Create a new communication endpoint
bind	Attach a local address to a socket
listen	Announce willingness to accept connections
accept	Block caller until a connection request arrives
connect	Actively attempt to establish a connection
write	Send(write) some data over the connection
read	Receive(read) some data over the connection
close	Release the connection

1.28 TCP communications pattern

TCP communications pattern



1.29 TCP Server fork

TCP Server fork

- server starts loop
 - blocks on accept for connection from client
- after accept:
 - accept returns dedicated connection socket
 - server forks to service client request
- parent process
 - closes dedicated connection socket
 - continues to block for next accept
- child process
 - serves client request
 - communicates with client via dedicated connection socket

1.30 Job Control Terminology

Job Control Terminology

- Foreground job:
 - a job that has our immediate attention
 - user has to wait for job to complete
- Background job:
 - a job that the user does not wait for
 - it runs independently of user interaction
- Unix shells allow users to:
 - make jobs execute in the background,
 - move jobs from foreground to background,
 - determine their status, and terminate them
- to execute command in the background, put **&** after it

1.31 Managing jobs

Managing jobs

- display jobs
 - command “jobs” lists your active jobs
 - each job has job number
 - job number with “%” is used to refer to job
- send job to background
 - bg
- move job to foreground
 - fg

1.32 Signaling jobs

Signaling jobs

- command to send signal to job:

kill

Examples:

kill -HUP 12324

kill -INT %1

1.33 Scheduling Utilities

Scheduling Utilities

- crontab
 - run a job based on a schedule
 - job is executed on a periodic basis
- at
 - run a job some time in the future
- batch
 - run a job when system load is low

1.34 Administration

Administration

- User Management
 - sudo
- Software Management
 - apt-get, synaptic
- File system Management
 - fdisk, mkfs, mount, fsck

1.35 Review Summary

Review Summary

- General UNIX knowledge
- C++ programming: IO management
- Systems programming: process & pipe
- Networking concepts
- UDP programming
- TCP programming
- System administration