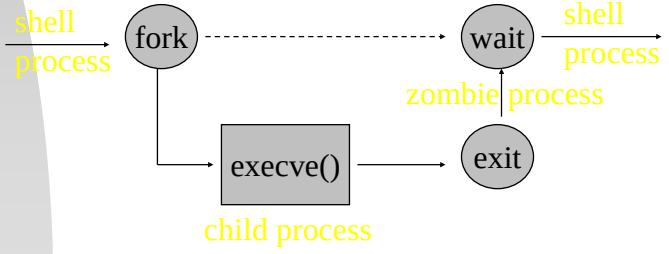
# Chp1 Introduction

- Objective
  - Briefly describe services provided by various versions of the UNIX operating system.
- Logging In
  - /etc/passwd – local machine or NIS DB
    - root:x:0:1:Super-User:/root:/bin/tcsh
    - Login-name, encrypted passwd, numeric user-ID, numeric group ID, comment, home dir, shell program
  - /etc/shadow – with "x" indicated for passwd



# Introduction

- Shell
  - Command interpreters
    - Built-in commands, e.g., umask
    - (External) commands, e.g., ls



# Introduction

- Shells
  - Bourne shell, /bin/sh
    - Steve Bourne at Bell Labs
  - C shell, /bin/csh
    - Bill Jay at Berkeley
      - Command-line editing, history, job-control, etc
  - KornShell. /bin/ksh
    - David Korn (successor of Bourne shell)
    - Command-line editing, job-control, etc
  - .cshrc

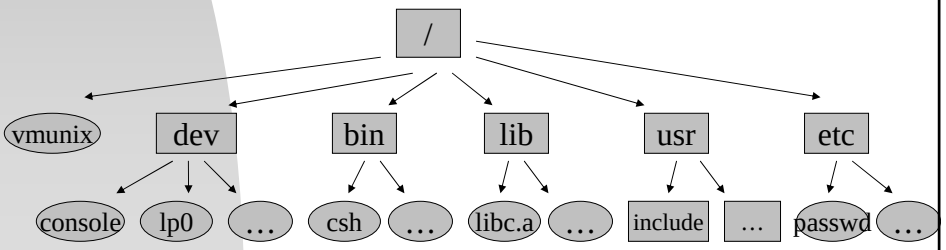


# Introduction

- Filesystem
  - A hierarchical arrangement of directories and files – starting in root /
- File
  - No / or null char in filenames
  - . and ..
  - BSD: 255-char filenames (14 in the past)
  - Attributes – stat()
    - Type, size, owner, permissions, modification/access time, etc.
- Directory
  - Files with directory entries, e.g., { (filenames, inode) }

# Introduction – Files and Dir

- File
  - A sequence of bytes
- Directory
  - A file that includes info on how to find other files.




* Use command “mount” to show all mounted file systems!



# Introduction

- Path name
  - Absolute path name
    - Start at the root / of the file system
    - /user/john/fileA
  - Relative path name
    - Start at the “current directory” which is an attribute of the process accessing the path name.
    - ./dirA/fileB
- Links
  - Symbolic Link – 4.3BSD
    - A file containing the path name of another file can across file-system boundaries. (home/prof/ktw>ln –s ../ktw ktw-test)
  - Hard Link
    - . or ..

# Introduction

- Directories
  - /vmunix - binary root image of UNIX
  - /dev - device special files, e.g., /dev/console
  - /bin - binaries of UNIX system programs
    - /usr/ucb - written by Berkley instead of AT&T
    - /usr/local/bin - written at the local site
  - /lib - library files, e.g., those for C
  - /user - directories for users, e.g., /user/john
  - /etc - administrative files and programs, e.g., passwd
  - /tmp - temporary files

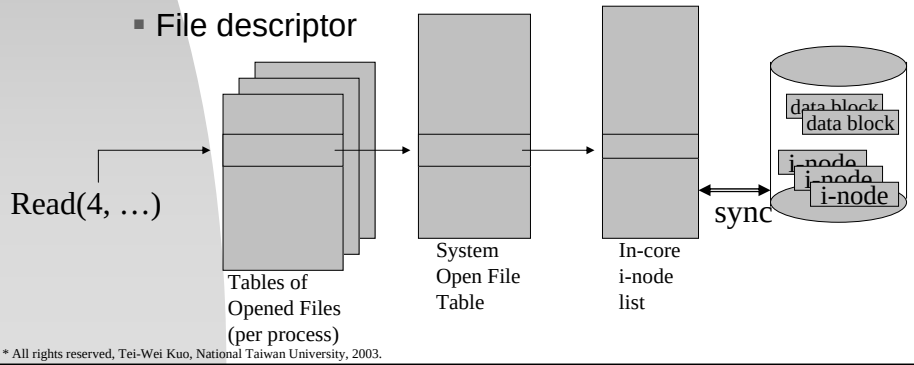


# Introduction

- Program 1.1 – Page 4
  - List all the files in a directory
- Note
  - “ourhdr.h”, err_sys() and err_quit() in Appendix B
  - opendir(), readdir(), closedir()
    - No ordering of directory entries
- Working directory
  - Goes with each process
- Home Directory

# Introduction – Input/Output

- Operations
  - open, close, read, write, trunc, lseek, dup, rename, chmod, chown, fcntl, ioctl, mkdir, cd, opendir, readdir, closedir, etc.
    - File descriptor






# Introduction

- File descriptor
  - Standard input (stdin), standard output (stdout), standard error (stderr)
  - Connected to the terminal if nothing special is done.
- I/O redirection
  - ls > file.list
- Unbuffered I/O: open, close, read, write, lseek
  - Program 1.2 – Page 7
  - Copies of stdin to stdout
  - STDIN_FILENO, STDOUT_FILENO in <unistd.h> , POSIX.1
  - ls | a.out > datafile

# Introduction

- Advantages of standard I/O functions such as fgets() and printf()
  - No need to worry about the optimal block size – a buffered interface
  - Handling of line input
- Misc
  - <stdio.h>
  - Program 1.3 – Page 9
    - Copy stdin to stdout using standard I/O



# Introduction

- Programs and Processes
  - Program – an executable file residing in a disk file
  - exec(), etc.
  - Process – an executing instance of a program
    - Unique Process ID
    - Program 1.4 – Page 10
      - Print process ID

# Introduction

- Process Control
  - Three primary functions: fork(), exec(), waitpid()
  - Program 1.5 – Page 11
    - Read commands from stdin and execute them
- Note
  - End-of-file: ^D
  - fork(), execlp(), waitpid()
  - No parsing of the input line
  - execlp(file, arg0 , …, argn, (char *) 0)
    - If file does not contain a slash character, the path prefix for this file is obtained by a search of the directories passed in the PATH environment variable.



# Introduction

- ANSI C Features
  - Function Prototypes
    - ssize_t read(int, void *, size_t);
    - void *malloc(size_t)
  - Generic Pointers
    - void * - avoid type casting
  - Primitive System Data Types
    - ssize_t, pid_t, etc.
    - <sys/types.h> included in <unistd.h>
    - Prevent programs from using specific data types – each implementation choose its proper data types by “typedef”!

# Introduction

- Error Handling
  - errno in <errno.h>  (sys/errno.h)
    - E.g., 15 error numbers for open()
    - #define ENOTTY  25      /* Inappropriate ioctl for device      */
    - Never cleared if no error occurs
    - No value 0 for any error number
- Functions
  - char *strerror (int errnum)  (<string.h>)
  - void perror(const char *msg) (<stdio.h>)
- Program 1.6 – Page 15
  - demo perror and strerror



# Introduction

- UNIX – A Layer Architecture
  - System Calls
    - Programmer Interface to UNIX
    - Trap 40 – VAX 4.2BSD
    - R0 – error code
  - Categories
    - File Manipulation
      - Devices are special files under “/dev”!
    - Process Control
    - Information Manipulation

# Introduction

- User Identification
  - Numeric value in /etc/passwd
    - 0 for root/superuser
  - Unchangeable and for access permission control
- Group ID
  - Numeric value in /etc/passwd
  - /etc/group
- Supplementary Group ID's
  - /etc/group (4.2BSD allows 16 additional groups.)
  - adm::4:root,adm,daemo
- "ls –l" uses /etc/passwd and /etc/group



# Introduction

- Signals
  - To notify a process that some condition has occurred
- Action
  - Ignore the signal
  - Execute the default action
    - E.g., for SIGFPE (divided by zero)
  - Provide a function
- Signal Generation
  - Terminal keys (^c ~> SIGINT), kill – owner-only
- Program 1.8 – Page 18
  - Read commands and exec + signal SIGINT

# Introduction

- Time Values
  - Calendar time
    - In seconds since the Epoch (00:00:00 January 1, 1970, Coordinated Universal Time, i.e., UTC)
    - type *time_t*
  - Process time
    - In clock ticks (divided by CLK_TCK -> secs)
    - type *clock_t*
    - Clock time, user/system CPU time

  > time grep _POSIX_SOURCE */*.h > /dev/null

  0.25u 0.25s 0:03.51 14.2%



# Introduction

- System Calls vs Library Functions
  - System Calls
    - 50 for Unix Ver 7, 110 for 4.3+BSD, 120 for SVR4
  - Unix Technique
    - Same function names for system calls
- Differences
  - Fixed set, more elaborate functionality
    - malloc() calls sbrk() → better allocated space management
    - gmtime() calls time() → seconds into broken-down time!
    - fgets() calls read() → unbuffered I/O -> buffered I/O
- Misc
  - Process control: fork(), exec(), wait() invoked directly from application code (vs system()).