Operating Systems

Vlastimil Babka

Lubomír Bulej

Vojtěch Horký

Petr Tůma

Operating Systems

by Vlastimil Babka, Lubomír Bulej, Vojtěch Horký, and Petr Tůma

This material is a work in progress that is provided on a fair use condition to support the Charles University Operating Systems lecture. It should not be used for any other purpose than to support the lecture. It should not be copied to prevent existence of outdated copies. It comes without warranty of any kind.

This is version 1854246b129de719fb64f40976b04a31545a422e (modified) generated on 2019-09-30 17:18:35.
For the latest version, check http://d3s.mff.cuni.cz/~ceres.

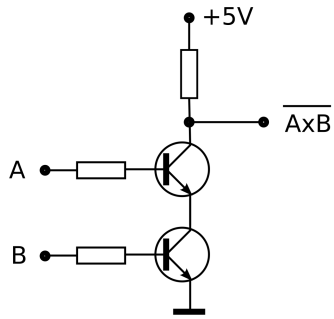
Table of Contents

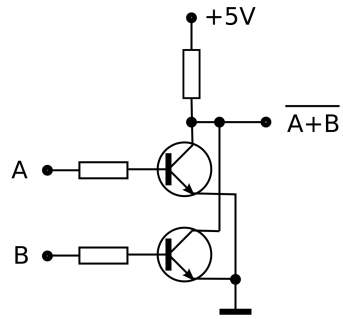
Chapter 1. Introduction

Basic Concepts

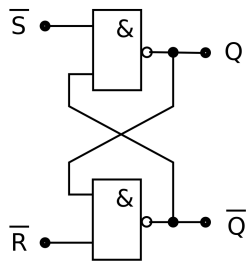
Hardware Building Blocks

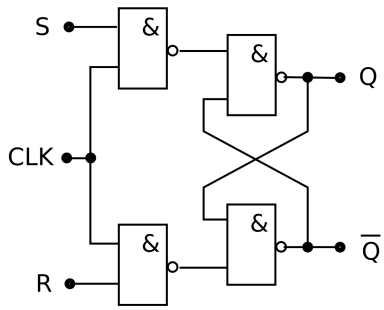
Composing NAND And NOR Gates





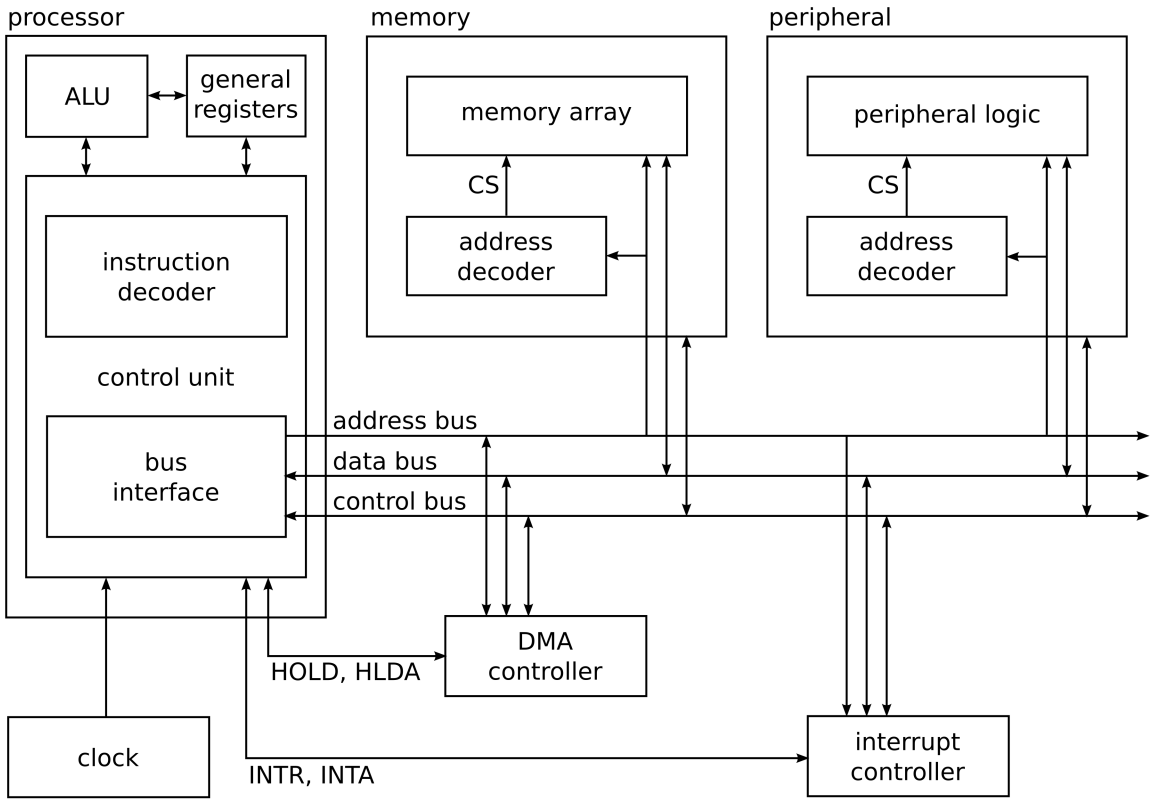
Composing RS Flip Flops





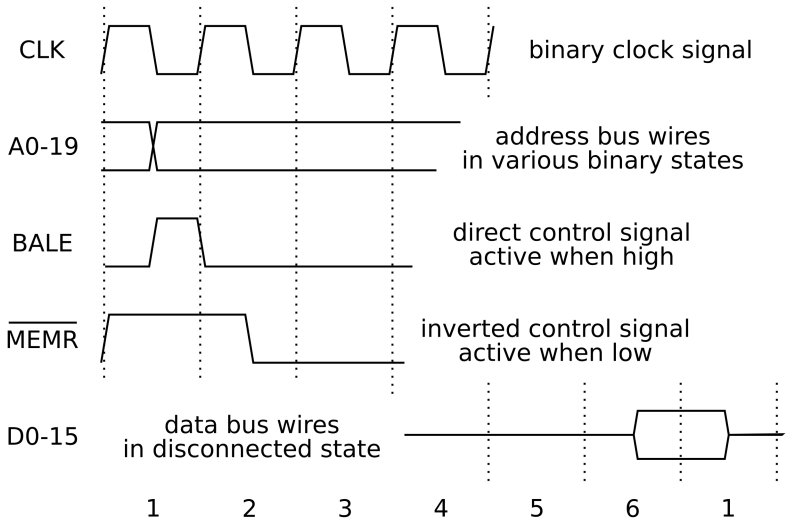
Basic Computer Architecture

Basic Computer Architecture Example



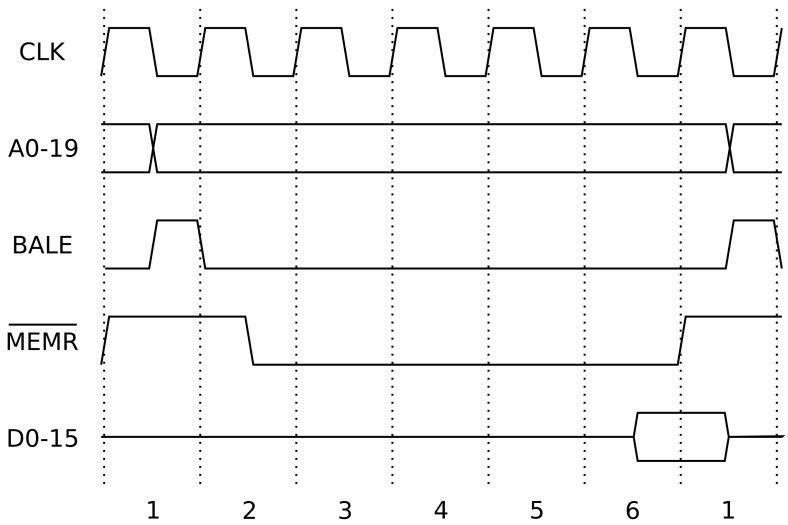
Processor Bus

Timing Diagram Example

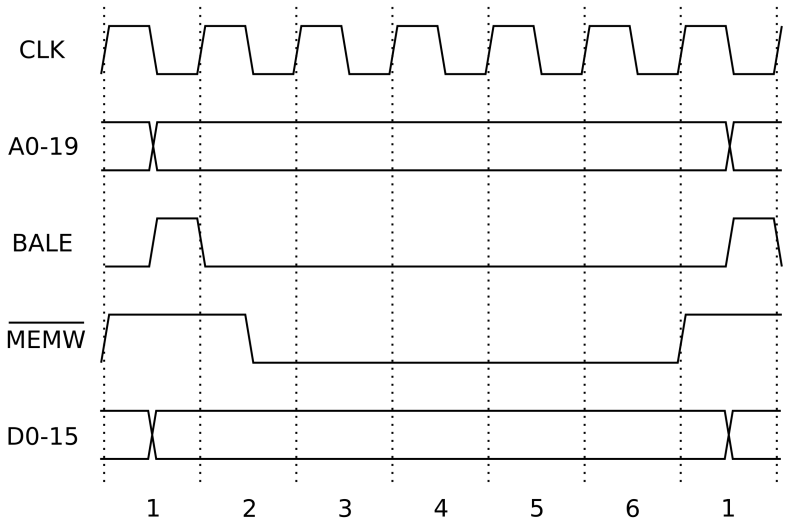


Example: ISA Bus

ISA Bus Read Cycle



ISA Bus Write Cycle

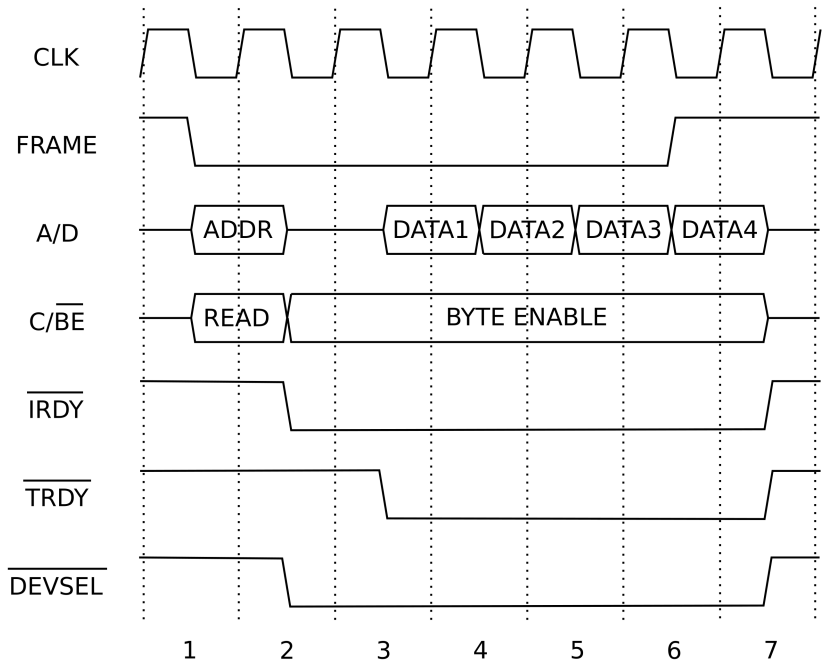


Advances In Bus Architecture

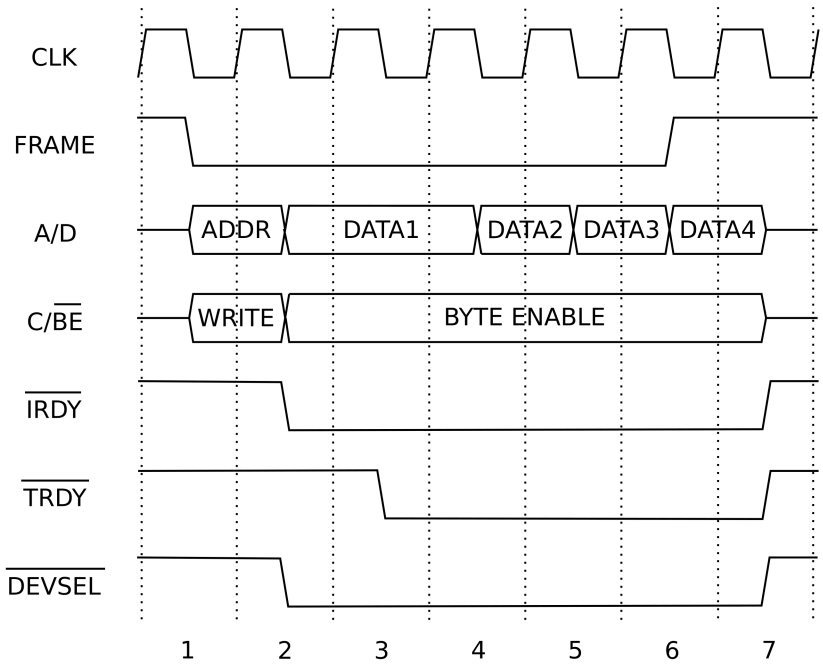
Burst Access

Example: PCI Bus

PCI Bus Read Cycle

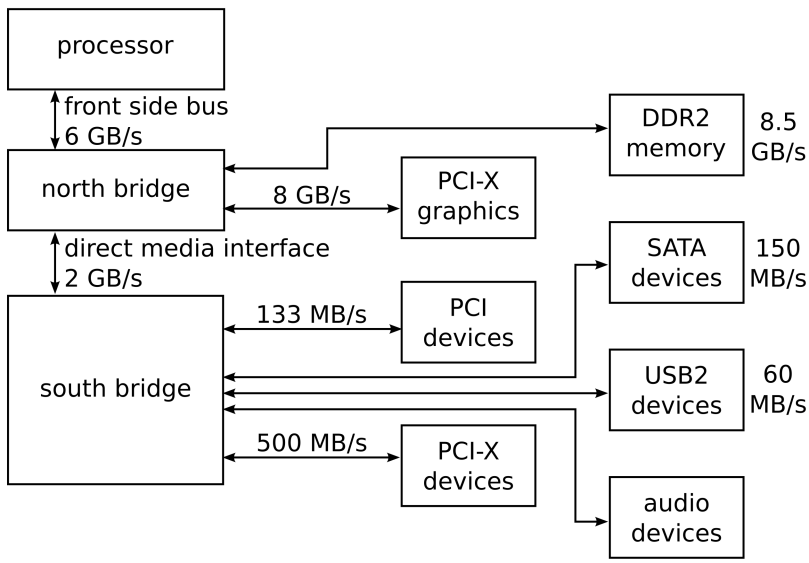


PCI Bus Write Cycle



Multiple Buses

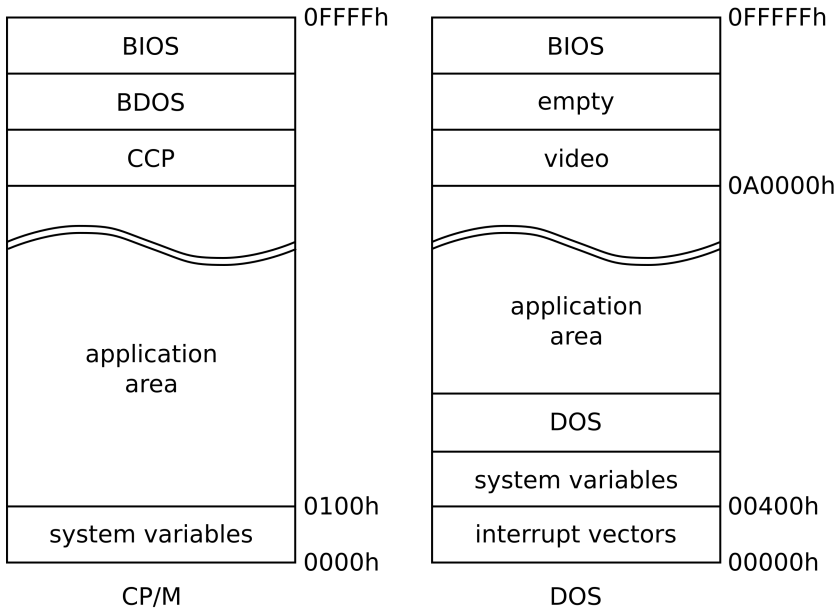
Multiple Buses Example



Operating System Structure

Monolithic Systems

Simple Monolithic Operating Systems Example



Chapter 2. Process Management

Process Alone

Starting A Process

Bootstrapping

Example: Booting MSIM

Simulator Bootstrap Example

Simulator configuration fragment.

```
add rom loadermem 0x1FC00000
loadermem generic 4K
loadermem load "kernel/loader.bin"
```

Loader entry code fragment.

```
.globl __start
.ent   __start

__start:

        la $ra, 0x80000400      ;hardcoded kernel entry address
        j $ra                   ;jump
        nop                     ;branch delay slot

.end __start
```

Kernel entry code fragment.

```
.section .excvec, "ax"          ;emit code into .execvec section
                                ;flags say allocatable and executable

.org   0x400                    ;hardcoded kernel entry address
.globl start
.ent   start

start:

        la $sp, 0x80000400      ;hardcoded stack pointer address
        jal main                ;jump and link
        nop                     ;branch delay slot

        halt                    ;a macro to stop the simulator

.end start
```

Linker script fragment.

```
SECTIONS {
    .kernel 0x80000000 : {      /* output section kernel with address */
        *(.excvec)              /* input section .execvec goes first */
        *(.text .text.*)        /* .text sections come next */
        *(.rodata .rodata.*)    /* .rodata sections next */
        ...
```

Relocating

Absolute Addressing Example

Declaring and accessing a global variable in C.

```
static int i;           // declare a global variable
...
i = 0x12345678;         // access the global variable
```

The C code compiled into Intel 80x86 assembler.

```
.comm i,4,4             ;declare i as 4 bytes aligned at 4 bytes boundary
...
movl $0x12345678,i      ;write value 12345678h into target address i
```

The assembler code compiled into Intel 80x86 machine code.

```
C705                    ;movl
C0950408                ;target address 080495C0h
78563412                ;value 12345678h
```

Relative Addressing Example

Declaring and accessing a global variable in C.

```
static int i;           // declare a global variable
...
i = 0;                  // access the global variable
```

The C code compiled into position independent Intel 80x86 assembler.

```
.comm i,4,4             ;declare i as 4 bytes aligned at 4 bytes boundary
...
call __get_thunk        ;get program starting address in ECX
addl $_GOT_,%ecx        ;calculate address of global table of addresses in ECX
movl $0,i@GOT(%ecx)     ;write value 0 into target address i relative from ECX
```

The assembler code compiled into position independent Intel 80x86 machine code.

```
E8                      ;call
1C000000                ;target address 0000001Ch bytes away from here
81C1                    ;addl target ECX
D9110000                ;value 000011D9h
C781                    ;movl target address relative from ECX
20000000                ;target address 00000020h bytes away from ECX
00000000                ;value 00000000h
```

Example: Program Image In Intel HEX

Intel HEX Format

```
:LLAAAATTxxxxCC
```

- LL - length of the data
- AAAA - address of the data in memory
- TT - indication of last line

- xxxx - data
- CC - checksum of the data

Example: Program Image In DOS

DOS EXE Format

```
Offset  Length  Contents
------------------------------------------------------------------------
00h     2       Magic (0AA55h)
02h     2       Length of last block
04h     2       Length of file in 512B blocks (L)
06h     2       Number of relocation table entries (R)
08h     2       Length of header in 16B blocks (H)
0Ah     2       Minimum memory beyond program image in 16B blocks
0Ch     2       Maximum memory beyond program image in 16B blocks
0Eh     4       Initial stack pointer setting (SS:SP)
12h     2       File checksum
14h     4       Initial program counter setting (CS:IP)
18h     2       Offset of relocation table (1Ch)
1Ah     2       Overlay number
1Ch     R*4h    Relocation table entries
H*10h   L*200h  Program image
```

Linking

Example: Executable And Linking Format

Headers

ELF Executable Header Example

```
> readelf --file-header /bin/bash

ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF64
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           Advanced Micro Devices X86-64
  Version:                           0x1
  Entry point address:               0x41d238
  Start of program headers:          64 (bytes into file)
  Start of section headers:          964960 (bytes into file)
  Flags:                             0x0
  Size of this header:               64 (bytes)
  Size of program headers:           56 (bytes)
  Number of program headers:         10
  Size of section headers:           64 (bytes)
  Number of section headers:         32
  Section header string table index: 31
```

ELF Library Header Example

```
> readelf --file-header /lib/libc.so.6

ELF Header:
  Magic:   7f 45 4c 46 01 01 01 03 00 00 00 00 00 00 00 00
  Class:                             ELF32
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - GNU
  ABI Version:                       0
  Type:                              DYN (Shared object file)
  Machine:                           Intel 80386
  Version:                           0x1
  Entry point address:               0x44564790
  Start of program headers:          52 (bytes into file)
  Start of section headers:          2009952 (bytes into file)
  Flags:                             0x0
  Size of this header:               52 (bytes)
  Size of program headers:           32 (bytes)
  Number of program headers:         10
  Size of section headers:           40 (bytes)
  Number of section headers:         43
  Section header string table index: 42
```

Sections

ELF Sections Example

```
> readelf --sections /lib/libc.so.6

There are 43 section headers, starting at offset 0x1eab60:

Section Headers:
  [Nr] Name              Type            Addr     Off    Size   ES Flg Lk Inf Al
...
  [ 9] .rel.dyn          REL             4455f3e4 0143e4 002a18 08   A  4   0  4
  [10] .rel.plt          REL             44561dfc 016dfc 000058 08   A  4  11  4
  [11] .plt              PROGBITS        44561e60 016e60 0000c0 04  AX  0   0 16
  [12] .text             PROGBITS        44561f20 016f20 14010c 00  AX  0   0 16
...
  [32] .data             PROGBITS        446f9040 1ad040 000e7c 00  WA  0   0 32
  [33] .bss              NOBITS          446f9ec0 1adebc 002bfc 00  WA  0   0 32
  [34] .comment          PROGBITS        00000000 1adebc 00002c 01  MS  0   0  1
  [35] .note.stapsdt     NOTE            00000000 1adee8 0002c4 00      0   0  4
  [36] .symtab           SYMTAB          00000000 1ae1ac 021880 10     37 6229  4
  [37] .strtab           STRTAB          00000000 1cfa2c 01a786 00      0   0  1
...
Key to Flags:
  W (write), A (alloc), X (execute), M (merge), S (strings)
  I (info), L (link order), G (group), T (TLS), E (exclude), x (unknown)
  O (extra OS processing required) o (OS specific), p (processor specific)
```

ELF Relocations Example

```
> readelf --relocs /lib/libc.so.6

Relocation section '.rel.dyn' at offset 0x134e4 contains 1457 entries:
 Offset     Info    Type            Sym.Value  Sym. Name
436f71b0  00000008 R_386_RELATIVE
436f8e74  0000000e R_386_TLS_TPOFF
436f8e90  00058206 R_386_GLOB_DAT    436f9d7c   stderr
436f9008  0004de07 R_386_JUMP_SLOT   435c3b70   malloc
...
```

Segments

ELF Segments Example

```
> readelf --segments /bin/bash

Elf file type is EXEC (Executable file)
Entry point 0x41d238
There are 10 program headers, starting at offset 64

Program Headers:
  Type           Offset             VirtAddr           PhysAddr
                 FileSiz            MemSiz              Flags  Align
  PHDR           0x0000000000000040 0x0000000000400040 0x0000000000400040
                 0x0000000000000230 0x0000000000000230  R E    8
  INTERP         0x0000000000000270 0x0000000000400270 0x0000000000400270
                 0x000000000000001c 0x000000000000001c  R      1
      [Requesting program interpreter: /lib64/ld-linux-x86-64.so.2]
  LOAD           0x0000000000000000 0x0000000000400000 0x0000000000400000
                 0x00000000000d9cd4 0x00000000000d9cd4  R E    200000
  DYNAMIC        0x00000000000d9df0 0x00000000006d9df0 0x00000000006d9df0
                 0x00000000000001f0 0x00000000000001f0  RW     8
  STACK          0x0000000000000000 0x0000000000000000 0x0000000000000000
                 0x0000000000000000 0x0000000000000000  RW     8
...

 Section to Segment mapping:
  Segment Sections...
   00
   01     .interp
   02     .interp .note .dynsym .rela .init .fini .plt .text ...
...
```

ELF Dynamic Information Example

```
> readelf --dynamic /bin/bash

Dynamic section at offset 0xd9df0 contains 30 entries:
  Tag        Type                         Name/Value
 0x0000000000000001 (NEEDED)             Shared library: [libtinfo.so.5]
 0x0000000000000001 (NEEDED)             Shared library: [libdl.so.2]
 0x0000000000000001 (NEEDED)             Shared library: [libc.so.6]
 0x000000000000000c (INIT)               0x41adb0
 0x000000000000000d (FINI)               0x4a6374
 0x0000000000000019 (INIT_ARRAY)         0x6d9dd8
 0x0000000000000005 (STRTAB)             0x8e2b30
 0x0000000000000006 (SYMTAB)             0x403988
...
```

Calling Operating System

Example: CP/M System Call Interface

CP/M BDOS System Call Example

```
ReadKey:    mvi     c,1         ; keyboard read service
            call    5           ; call BDOS entry point
            cpi     a,0Dh       ; is returned key code ENTER ?
            jnz     ReadKey     ; repeat keyboard read until it is
```

CP/M BIOS System Call Entry Points

```
jmp     BOOT            ;cold boot
jmp     WBOOT           ;warm boot
jmp     CONST           ;console status
jmp     CONIN           ;console input
...
jmp     HOME            ;disk head to track 0
...
jmp     SETDMA          ;set memory transfer address
jmp     READ            ;read sector
jmp     WRITE           ;write sector
```

Example: Linux System Call API On Intel 80x86

Library System Call Example

```
ssize_t read (int fd, void *buf, size_t count);
...
int hFile;
ssize_t iCount;
char abBuffer [1024];
iCount = read (hFile, &abBuffer, sizeof (abBuffer));

pushl $1024             ;sizeof (abBuffer)
pushl $abBuffer         ;&abBuffer
pushl hFile             ;hFile
call  read              ;call the library
addl  $12,%esp          ;remove arguments from stack
movl  %eax,iCount       ;save result
```

Linux Gate Library Based On INT 80h

```
__kernel_vsyscall: int $0x80
                   ret
```

Linux Gate Library Based On SYSENTER And SYSEXIT

```
__kernel_vsyscall: push %ecx
                   push %edx
                   push %ebp
__resume:          mov  %esp,%ebp
                   sysenter
```

```
                    jmp  __resume        ;hack for syscall resume

__return:           pop  %ebp            ;this is where
                    pop  %edx             ;the SYSEXIT
                    pop  %ecx              ;returns
                    ret
```

Example: Linux Syslet API

Syslet Atom Structure

```
struct syslet_uatom
{
  u32 flags;
  u32 nr;               // what syscall to execute
  u64 ret_ptr;          // where to store return value
  u64 next;             // what is the next atom in the chain
  u64 arg_ptr [6];      // what are the syscall arguments
  u64 private;          // free for application use
};
```

- SYSLET_STOP_ON_ZERO - termination condition value
- SYSLET_STOP_ON_NONZERO
- SYSLET_STOP_ON_NEGATIVE
- SYSLET_SKIP_TO_NEXT_ON_STOP - conditional branching in atom array

Syslet Usage Example

```
struct request
{
  u64 filename_ptr;
  u64 fd;

  struct syslet_uatom open_file;
  struct syslet_uatom read_file;
  struct syslet_uatom close_file;
};

request req;

req.open_file.nr = __NR_sys_open;
req.open_file.arg_ptr [0] = (u64) &req.filename_ptr;
...
req.open_file.ret_ptr = (u64) &req.fd;
req.open_file.flags = (u64) SYSLET_STOP_ON_NEGATIVE;
req.open_file.next = (u64) &req.read_file;
...

struct syslet_uatom *done;

done = sys_async_exec (&req.open_file ...);
```

Example: Windows System Call API On Intel 80x86

Library System Call Example

```
int MessageBox (
    HWND hwndOwner,
    LPCTSTR lpszText,
    LPCTSTR lpszTitle,
    UINT fuStyle);
...
MessageBox (0, zMessageText, zWindowTitle, MB_OK || MB_SYSTEMMODAL || MB_ICONHAND);

push MBOK or MB_SYSTEMMODAL or MB_ICONHAND
push offset zWindowTitle
push offset zMessageTest
push 0
call MessageBoxA                ;call the library
add  esp,16                     ;remove arguments from stack
```

What Is The Interface

Example: Posix Process And Thread API

Posix Process Creation System Calls

```
pid_t fork (void);
int execve (const char *filename, char *const argv [], char *const envp []);

pid_t wait (int *status);
pid_t waitpid (pid_t pid, int *status, int options);

void exit (int status);
```

Posix Thread Creation System Calls

```
int pthread_create (
  pthread_t *thread,
  pthread_attr_t *attr,
  void * (*start_routine) (void *),
  void *arg);

int pthread_join (
  pthread_t thread,
  void **return_value);

void pthread_exit (
  void *return_value);

int pthread_detach (
  pthread_t thread);
```

Posix Thread Specific Data Calls

```
int pthread_key_create (
  pthread_key_t *key,
  void (* destructor) (void *));

int pthread_setspecific (
  pthread_key_t key,
  const void *value);
void *pthread_getspecific (
  pthread_key_t key);
```

Example: Windows Process And Thread API

Windows Process Creation System Calls

```
BOOL CreateProcess (
  LPCTSTR lpApplicationName,
  LPTSTR lpCommandLine,
  LPSECURITY_ATTRIBUTES lpProcessAttributes,
  LPSECURITY_ATTRIBUTES lpThreadAttributes,
  BOOL bInheritHandles,
  DWORD dwCreationFlags,
  LPVOID lpEnvironment,
  LPCTSTR lpCurrentDirectory,
  LPSTARTUPINFO lpStartupInfo,
  LPPROCESS_INFORMATION lpProcessInformation
);

VOID ExitProcess (
  UINT uExitCode);

DWORD WaitForSingleObject (
  HANDLE hHandle,
  DWORD dwMilliseconds
);
```

Windows Thread Creation System Calls

```
HANDLE CreateThread (
  LPSECURITY_ATTRIBUTES lpThreadAttributes,
  SIZE_T dwStackSize,
  LPTHREAD_START_ROUTINE lpStartAddress,
  LPVOID lpParameter,
  DWORD dwCreationFlags,
  LPDWORD lpThreadId
);

VOID ExitThread (
  DWORD dwExitCode);
```

Windows Fiber Creation System Calls

```
LPVOID ConvertThreadToFiber (
  LPVOID lpParameter);

LPVOID CreateFiber (
  SIZE_T dwStackSize,
  LPFIBER_START_ROUTINE lpStartAddress,
  LPVOID lpParameter);

VOID SwitchToFiber (
  LPVOID lpFiber);

VOID DeleteFiber (
  LPVOID lpFiber);
```

Windows Thread Local Data Calls

```
DWORD TlsAlloc (void);
BOOL TlsFree (
  DWORD dwTlsIndex);

BOOL TlsSetValue (
  DWORD dwTlsIndex,
  LPVOID lpTlsValue);
LPVOID TlsGetValue (
  DWORD dwTlsIndex);
```

Windows Fiber Local Data Calls

```
DWORD FlsAlloc (
  PFLS_CALLBACK_FUNCTION lpCallback);
BOOL FlsFree (
  DWORD dwFlsIndex);

BOOL FlsSetValue (
  DWORD dwFlsIndex,
  PVOID lpFlsValue);
PVOID FlsGetValue (
  DWORD dwFlsIndex);
```

Windows Stack Guarantee Call

```
BOOL SetThreadStackGuarantee (
  PULONG StackSizeInBytes);
```

Example: Linux Clone API

Linux Clone Call

```
int clone (
  int (*fn) (void *),
  void *child_stack,
  int flags,
  void *arg,
  ...);
```

- CLONE_SIGHAND - share signal handler table
- CLONE_FILES - share file descriptor table
- CLONE_FS - share filesystem context (root directory, current directory, file creation mask)
- CLONE_IO - share device scheduler context
- CLONE_VM - share address space

- CLONE_NEWIPC - create new process communication namespace
- CLONE_NEWPID - create new process identifier namespace
- CLONE_NEWUTS - create new host platform namespace
- CLONE_NEWNET - create new network namespace
- CLONE_NEWNS - create new mount namespace

Example: Posix Dynamic Linker API

Posix Dynamic Linker Calls

```
void *dlopen (
  const char *filename,
  int flag);
int dlclose (
  void *handle);
```

- RTLD_LAZY - resolve symbols on use when possible
- RTLD_GLOBAL - make symbols available to other libraries

```
void *dlsym (
  void *handle,
  const char *symbol);
```

- RTLD_DEFAULT - use default symbol lookup order
- RTLD_NEXT - lookup in libraries that follow this

Example: Java Thread API

Java Thread Class

```
class java.lang.Thread implements java.lang.Runnable {
  java.lang.Thread ();
  java.lang.Thread (java.lang.Runnable);

  void start ();
  void run ();
  void interrupt ();
  boolean isInterrupted ();

  void join () throws java.lang.InterruptedException;
  void setDaemon (boolean);
```

```
  boolean isDaemon ();

  static java.lang.Thread currentThread ();
  static void yield ();
  static void sleep (long) throws java.lang.InterruptedException;

  ...
}
```

Example: OpenMP Thread API

OpenMP Thread Creation Directives

```
#pragma omp parallel private (iThreads, iMyThread)
{
  iThreads = omp_get_num_threads ();
  iMyThread = omp_get_thread_num ();
  ...
}

#pragma omp parallel for
  for (i = 0 ; i < MAX ; i ++)
    a [i] = 0;

#pragma omp parallel sections
{
  #pragma omp section
    DoOneThing ();
  #pragma omp section
    DoAnotherThing ();
}
```

Achieving Parallelism

Multiprocessing On Uniprocessors

Processor State

Example: Linux Processor Context Switching

Linux 2.6.8 80x86 Processor Context Switching

```
#define __SAVE_ALL \
        cld; \
        pushl %es; \
        pushl %ds; \
        pushl %eax; \
        pushl %ebp; \
        pushl %edi; \
        pushl %esi; \
        pushl %edx; \
        pushl %ecx; \
```

```
        pushl %ebx; \
        movl $(__USER_DS), %edx; \
        movl %edx, %ds; \
        movl %edx, %es;

#define __RESTORE_INT_REGS \
        popl %ebx;      \
        popl %ecx;      \
        popl %edx;      \
        popl %esi;      \
        popl %edi;      \
        popl %ebp;      \
        popl %eax

#define __RESTORE_REGS  \
        __RESTORE_INT_REGS; \
111:    popl %ds;       \
222:    popl %es;       \
.section .fixup,"ax";   \
444:    movl $0,(%esp); \
        jmp 111b;       \
555:    movl $0,(%esp); \
        jmp 222b;       \
.previous;              \
.section __ex_table,"a";\
        .align 4;       \
        .long 111b,444b;\
        .long 222b,555b;\
.previous

#define __RESTORE_ALL   \
        __RESTORE_REGS  \
        addl $4, %esp;  \
333:    iret;           \
.section .fixup,"ax";   \
666:    sti;            \
        movl $(__USER_DS), %edx; \
        movl %edx, %ds; \
        movl %edx, %es; \
        pushl $11;      \
        call do_exit;   \
.previous;              \
.section __ex_table,"a";\
        .align 4;       \
        .long 333b,666b;\
.previous

#define SAVE_ALL \
        __SAVE_ALL; \
        __SWITCH_KERNELSPACE;

#define RESTORE_ALL \
        __SWITCH_USERSPACE; \
        __RESTORE_ALL;
```

Example: Kalisto Processor Context Switching

Kalisto MIPS Processor Context Switching

```
.macro SAVE_REGISTERS base

        sw $zero, REGS_OFFSET_ZERO(\base)

        sw $at,   REGS_OFFSET_AT(\base)

        sw $v0,   REGS_OFFSET_V0(\base)
        sw $v1,   REGS_OFFSET_V1(\base)

        sw $a0,   REGS_OFFSET_A0(\base)
        sw $a1,   REGS_OFFSET_A1(\base)
        sw $a2,   REGS_OFFSET_A2(\base)
        sw $a3,   REGS_OFFSET_A3(\base)

        ...

        sw $gp,   REGS_OFFSET_GP(\base)
        sw $fp,   REGS_OFFSET_FP(\base)
        sw $ra,   REGS_OFFSET_RA(\base)

.endm SAVE_REGISTERS


.macro LOAD_REGISTERS base

        lw $ra,   REGS_OFFSET_RA(\base)
        lw $fp,   REGS_OFFSET_FP(\base)
        lw $gp,   REGS_OFFSET_GP(\base)

        ...

        lw $a3,   REGS_OFFSET_A3(\base)
        lw $a2,   REGS_OFFSET_A2(\base)
        lw $a1,   REGS_OFFSET_A1(\base)
        lw $a0,   REGS_OFFSET_A0(\base)

        lw $v1,   REGS_OFFSET_V1(\base)
        lw $v0,   REGS_OFFSET_V0(\base)

        lw $at,   REGS_OFFSET_AT(\base)

        lw $zero, REGS_OFFSET_ZERO(\base)

.endm LOAD_REGISTERS


switch_cpu_context:

        /* Allocate a frame on the stack of the old thread and update
           the address of the stack top of the old thread. */

        addiu $sp, -CONTEXT_SIZE        ;Allocate space on stack
        sw $sp, ($a0)                   ;Save the old stack

        SAVE_REGISTERS $sp              ;Save general registers

        mflo $t0                        ;Few other registers that
        mfhi $t1                         ;the macro does not handle
        sw $t0, REGS_OFFSET_LO($sp)       ;need to be saved as well
        sw $t1, REGS_OFFSET_HI($sp)
```

```
mfc0 $t0, $status
sw $t0, REGS_OFFSET_STATUS($sp)
la $t1, ~CP0_STATUS_IE_MASK
and $t0, $t1
mtc0 $t0, $status               ;Disable interrupts

lw $sp, ($a1)                   ;Switch to the new stack

lw $t0, REGS_OFFSET_LO($sp)     ;Restore the registers in
lw $t1, REGS_OFFSET_HI($sp)      ;roughly the opposite
mtlo $t0                          ;order to fit the
mthi $t1                           ;stack semantics

LOAD_REGISTERS $sp

lw $k0, REGS_OFFSET_STATUS($sp)

addiu $sp, CONTEXT_SIZE         ;Free space on stack

j $ra                           ;Return to the newly
mtc0 $k0, $status                ;restored context
```

How To Decide Who Runs

Scheduling Requirements

- Responsiveness - reacting in reasonable time.
- Predictability - scheduling in consistent manner.
- Turnaround - minimizing time to complete a single task.
- Throughput - maximizing number of completed tasks.
- Efficiency - maximizing resource utilization.
- Fairness - treating tasks equally.

Application Classes

- Interactive - mostly waits, needs quick reaction to events.
- Batch - mostly works, benefits from low trashing.
- Realtime - needs guaranteed timing.

Example: Solaris Scheduler

Solaris Real Time Dispatcher Configuration

```
> /usr/sbin/dispadmin -c RT -g
# Real Time Dispatcher Configuration
RES=1000

# TIME QUANTUM                    PRIORITY
# (rt_quantum)                      LEVEL
```

```
      1000                  #        0
...
      1000                  #        9
       800                  #       10
...
       800                  #       19
       600                  #       20
...
       100                  #       55
       100                  #       56
       100                  #       57
       100                  #       58
       100                  #       59
```

Solaris Time Sharing Dispatcher Configuration

```
> /usr/sbin/dispadmin -c TS -g
# Time Sharing Dispatcher Configuration
RES=1000

# ts_quantum  ts_tqexp  ts_slpret  ts_maxwait ts_lwait  PRIORITY LEVEL
       200         0        50           0        50        #     0
...
       200         0        50           0        50        #     9
       160         0        51           0        51        #    10
       160         1        51           0        51        #    11
       160         2        51           0        51        #    12
       160         3        51           0        51        #    13
       160         4        51           0        51        #    14
       160         5        51           0        51        #    15
       160         6        51           0        51        #    16
       160         7        51           0        51        #    17
       160         8        51           0        51        #    18
       160         9        51           0        51        #    19
       120        10        52           0        52        #    20
...
        40        45        58           0        59        #    55
        40        46        58           0        59        #    56
        40        47        58           0        59        #    57
        40        48        58           0        59        #    58
        20        49        59       32000        59        #    59
```

Example: Windows Scheduler

Windows Thread Priority Calculation

	Idle	Normal	High	Realtime
Idle	1	1	1	16
Lowest	2	6	11	22
Below Normal	3	7	12	23
Normal	4	8	13	24
Above Normal	5	9	14	25
Highest	6	10	15	26
Time Critical	15	15	15	31

Example: Linux Late 2.6.X Series Scheduler

CFS Scheduler Tunables

sched_min_granularity_ns
: The minimum time a task will run unless it blocks.

sched_latency_ns
: The turn around period of the run queue.

sched_wakeup_granularity_ns
: The minimum time a previously sleeping task will run unless it blocks.

Example: Advanced Linux Scheduling Features

Scheduler Control Groups Example

```
> tree -d /sys/fs/cgroup

/sys/fs/cgroup
|-- blkio
|-- cpu
|   `-- system
|       |-- acpid.service
|       |-- chronyd.service
|       |-- crond.service
|       |-- dbus.service
|       |-- httpd.service
|       |-- mdmonitor.service
...
|       |-- sshd.service
|       |-- udev.service
|       `-- upower.service
|-- devices
|-- memory
...
```

Control Groups Configuration Example

```
group singlecore {
  perm {
   admin {
    # can change settings (e.g. cpumask)
    # or reset statistics
    uid = root;
   }
   task {
    # can run the tasks
    uid = user;
   }
  }
  cpuset {
   # limit to single CPU
   cpuset.cpus = 3;
   cpuset.mems = 0;
  }
  cpuacct {
```

```
  }
}

cgexec -g cpuset:singlecore /home/user/some_script.sh
```

Scheduler Control Group Tunables

cpu.shares
: Processor share this group gets relative to other groups.

cpu.cfs_period_us
: Defines a period for the purpose of limiting the CFS scheduler bandwidth.

cpu.cfs_quota_us
: Limits the CFS scheduler to particular quota within each period above.

cpu.rt_period_us
: Defines a period for the purpose of limiting the RT scheduler bandwidth.

cpu.rt_runtime_us
: Limits the RT scheduler to particular quota within each period above.

What Is The Interface

Example: Linux Scheduler API

Linux Old Style Scheduler Calls

```
int getpriority (int which, int who);
int setpriority (int which, int who, int prio);
```

- PRIO_PROCESS - set or get process priority
- PRIO_PGRP - set priority of all in group, get highest
- PRIO_USER - set priority of all owned by user, get highest

Linux New Style Scheduler Calls

```
int sched_setscheduler (
  pid_t pid,
  int policy,
  const struct sched_param *param);
int sched_getscheduler (pid_t pid);

int sched_setparam (pid_t pid, const struct sched_param *param);
int sched_getparam (pid_t pid, struct sched_param *param);

struct sched_param
{
  ...
```

```
  int sched_priority;
  ...
};

int sched_yield (void);
```

- SCHED_OTHER - normal time sharing process
- SCHED_BATCH - processor intensive time sharing process
- SCHED_IDLE - time sharing process to be run very rarely
- SCHED_FIFO - static priority with unlimited quantum
- SCHED_RR - static priority with limited quantum

Linux Scheduler Calls

```
int sched_setaffinity (
  pid_t pid,
  size_t cpusetsize,
  cpu_set_t *mask);
int sched_getaffinity (
  pid_t pid,
  size_t cpusetsize,
  cpu_set_t *mask);
```

Example: Windows Scheduler API

Windows Scheduler Calls

```
BOOL SetPriorityClass (
  HANDLE hProcess,
  DWORD dwPriorityClass);
DWORD GetPriorityClass (
  HANDLE hProcess);

BOOL SetThreadPriority (
  HANDLE hThread,
  int nPriority);
int GetThreadPriority (
  HANDLE hThread);

BOOL SetProcessPriorityBoost (
  HANDLE hProcess,
  BOOL DisablePriorityBoost);
BOOL SetThreadPriorityBoost (
  HANDLE hThread,
  BOOL DisablePriorityBoost);

BOOL SetProcessAffinityMask (
  HANDLE hProcess,
  DWORD_PTR dwProcessAffinityMask);
DWORD_PTR SetThreadAffinityMask (
  HANDLE hThread,
  DWORD_PTR dwThreadAffinityMask);
```

Windows Thread Pool Calls

```
PTP_POOL CreateThreadpool (
  PVOID reserved);
VOID CloseThreadpool (
  PTP_POOL ptpp);

BOOL SetThreadpoolThreadMinimum (
  PTP_POOL ptpp,
  DWORD cthrdMic);
VOID SetThreadpoolThreadMaximum (
  PTP_POOL ptpp,
  DWORD cthrdMost);

VOID SubmitThreadpoolWork (
  PTP_WORK pwk);
```

Windows Process Timing Call

```
BOOL GetProcessTimes (
  HANDLE hProcess,
  LPFILETIME lpCreationTime,
  LPFILETIME lpExitTime,
  LPFILETIME lpKernelTime,
  LPFILETIME lpUserTime);
```

Process Communication

Shared Memory

Example: System V Shared Memory

ShmGet System Call

```
int shmget (key_t key, size_t size, int shmflg);
```

- IPC_PRIVATE - private key value
- IPC_CREAT - object with key can be created
- IPC_EXCL - object with key must not exist

```
void *shmat (int shmid, const void *shmaddr, int shmflg);
int shmdt (const void *shmaddr);

key_t ftok (const char *pathname, int proj_id);
```

Shared Memory Listing

```
> ipcs -m
key        shmid      owner      perms      bytes      nattch     status
0x00000000 12345      root       600        123456     2          dest
0x00000000 123456     root       600        234567     2          dest
0x00000000 1234567    nobody     777        345678     2          dest
```

Example: Windows Shared Memory

CreateFileMapping System Call

```
HANDLE CreateFileMapping (
  HANDLE hFile,
  LPSECURITY_ATTRIBUTES lpFileMappingAttributes,
  DWORD flProtect,
  DWORD dwMaximumSizeHigh,
  DWORD dwMaximumSizeLow,
  LPCTSTR lpName);
```

- PAGE_READONLY
- PAGE_READWRITE
- PAGE_WRITECOPY
- SEC_IMAGE

MapViewOfFile System Call

```
LPVOID MapViewOfFile (
  HANDLE hFileMappingObject, DWORD dwDesiredAccess,
  DWORD dwFileOffsetHigh, DWORD dwFileOffsetLow,
  DWORD dwNumberOfBytesToMap);

LPVOID MapViewOfFileEx (
  HANDLE hFileMappingObject, DWORD dwDesiredAccess,
  DWORD dwFileOffsetHigh, DWORD dwFileOffsetLow,
  DWORD dwNumberOfBytesToMap, LPVOID lpBaseAddress);
```

- FILE_MAP_READ
- FILE_MAP_WRITE
- FILE_MAP_COPY
- FILE_MAP_ALL_ACCESS

Message Passing

Example: Posix Signals

Standard Signals

Name	Number	Meaning
SIGHUP	1	Controlling terminal closed
SIGINT	2	Request for interrupt sent from keyboard
SIGQUIT	3	Request for quit sent from keyboard
SIGILL	4	Illegal instruction
SIGTRAP	5	Breakpoint instruction
SIGABRT	6	Request for abort
SIGBUS	7	Illegal bus cycle
SIGFPE	8	Floating point exception
SIGKILL	9	Request for kill
SIGUSR1	10	User defined signal 1
SIGSEGV	11	Illegal memory access
SIGUSR2	12	User defined signal 2
SIGPIPE	13	Broken pipe
SIGALRM	14	Timer alarm
SIGTERM	15	Request for termination
SIGTERM	16	Illegal stack access
SIGCHLD	17	Child process status changed
SIGCONT	18	Request to continue when stopped
SIGSTOP	19	Request to stop
SIGTSTP	20	Request for stop sent from keyboard
SIGTTIN	21	Input from terminal when on background
SIGTTOU	22	Output to terminal when on background

Signal Handler Registration System Call

```
typedef void (*sighandler_t) (int);

sighandler_t signal (int signum, sighandler_t handler);
```

- SIG_DFL - use default signal handler
- SIG_IGN - ignore the signal

```
struct sigaction
{
  void (*sa_handler) (int);
```

```
  void (*sa_sigaction) (int, siginfo_t *, void *);
  sigset_t sa_mask;
  int sa_flags;
}

struct siginfo_t
{
  int      si_signo;    // Signal number
  int      si_errno;    // Value of errno
  int      si_code;     // Additional signal code
  pid_t    si_pid;      // Sending process PID
  uid_t    si_uid;      // Sending process UID
  int      si_status;   // Exit value
  clock_t  si_utime;    // User time consumed
  clock_t  si_stime;    // System time consumed
  sigval_t si_value;    // Signal value
  int      si_int;      // Integer value sent with signal
  void *   si_ptr;      // Pointer value sent with signal
  void *   si_addr;     // Associated memory address
  int      si_fd;       // Associated file descriptor
}

int sigaction (int signum, const struct sigaction *act, struct sigaction *oldact);
```

- sa_handler - signal handler with limited arguments
- sa_sigaction - signal handler with complete arguments
- sa_mask - what other signals to mask while in signal handler

- SA_RESETHAND - restore default signal handler after one signal
- SA_NODEFER - allow recursive invocation of this signal handler
- SA_ONSTACK - use alternate stack for this signal handler

Signal Masking System Call

```
int sigprocmask (int how, const sigset_t *set, sigset_t *oset);
int pthread_sigmask (int how, const sigset_t *set, sigset_t *oset);
```

- SIG_BLOCK - add blocking to signals that are not yet blocked
- SIG_UNBLOCK - remove blocking from signals that are blocked
- SIG_SETMASK - replace existing mask

Signal Send System Call

```
int kill (pid_t pid, int sig);
int pthread_kill (pthread_t thread, int sig);

union sigval
{
  int sival_int;
  void *sival_ptr;
}

int sigqueue (pid_t pid, int sig, const union sigval value);
```

Example: System V Message Passing

MsgSnd And MsgRcv System Calls

```
int msgsnd (int que, message *msg, int len, int flags);
int msgrcv (int que, message *msg, int len, int type, int flags);
```

MsgGet System Call

```
int msgget (key_t key, int msgflg);
```

Process Synchronization

Means For Synchronization

Active Waiting

Naive Active Wait For Critical Section

```
while (bCriticalSectionBusy)
{
  // Active waiting cycle until the
  // bCriticalSectionBusy variable
  // becomes false
}
bCriticalSectionBusy = true;

// Code of critical section comes here
...

bCriticalSectionBusy = false;
```

Improved Active Wait For Critical Section

```
while (true)
{
  // Indicate the intent to enter the critical section
  bIWantToEnter = true;
  // Enter the critical section if the other
  // process does not indicate the same intent
  if (!bHeWantsToEnter) break;
  // Back off to give the other process
  // a chance and continue the active
  // waiting cycle
  bIWantToEnter = false;
}

// Code of critical section comes here
...

bIWantToEnter = false;
```

Dekker Algorithm

```
// Indicate the intent to enter the critical section
bIWantToEnter = true;
while (bHeWantsToEnter)
{
  // If the other process indicates the same intent and
  // it is not our turn, back off to give the other
  // process a chance
  if (iWhoseTurn != MY_TURN)
  {
    bIWantToEnter = false;
    while (iWhoseTurn != MY_TURN) { }
    bIWantToEnter = true;
  }
}

// Code of critical section comes here
...

iWhoseTurn = HIS_TURN;
bIWantToEnter = false;
```

Peterson Algorithm

```
// Indicate the intent to enter the critical section
bIWantToEnter = true;
// Be polite and act as if it is not our
// turn to enter the critical section
iWhoseTurn = HIS_TURN;
// Wait until the other process either does not
// intend to enter the critical section or
// acts as if its our turn to enter
while (bHeWantsToEnter && (iWhoseTurn != MY_TURN)) { }

// Code of critical section comes here
...

bIWantToEnter = false;
```

Active Wait For Critical Section Using Atomic Swap

```
while (AtomicSwap (bCriticalSectionBusy, true))
{
  // Active waiting cycle until the
  // value of the bCriticalSectionBusy
  // variable has changed from false to true
}

// Code of critical section comes here
...

bCriticalSectionBusy = false;
```

Passive Waiting

Naive Passive Wait For Critical Section

```
if (AtomicSwap (bCriticalSectionBusy, true))
{
  // The critical section is busy, put
  // the process into the waiting queue
  oWaitingProcesses.Put (GetCurrentProcess ());
  // Wait until somebody wakes the process
  Sleep ();
}

// Code of critical section comes here
...

// See if any process is waiting in the queue
oWaitingProcess = oWaitingProcesses.Get ();

if (oWaitingProcess)
{
  // A process was waiting, let it enter the critical section
  Wake (oWaitingProcess);
}
else
{
  // No process was waiting, mark the critical section as free
  bCriticalSectionBusy = false;
}
```

Memory Models

Example: Invalid Register Optimization

```
// Assume a hot loop worth optimizing.
while (...)
{
  ...
  // Protect access to x if multithreaded.
  if (bThreaded) oLock.lock ();

  // Do a lot of work with x here.
  x = f (x);

  // Protect access to x if multithreaded.
  if (bThreaded) oLock.unlock ();
  ...
}

// Since x is used a lot it is kept in some register.
register = x;
while (...)
{
  ...
  if (bThreaded)
  {
    // External call may use x so it is written back.
    x = register;
    oLock.lock ();
    register = x;
```

```
  }

  // A lot of work done efficiently with x in register.
  register = f (register);

  if (bThreaded)
  {
    // External call may use x so it is written back.
    x = register;
    oLock.unlock ();
    register = x;
  }
  ...
}
x = register;
```

Example adjusted from literature, see references.

Example: Invalid Branch Optimization

```
// If x is 0 then set y to 0.
// Otherwise leave y unchanged.
if (x == 0) y = 0;

// If x is 0 then set y to 0.
// Otherwise leave y unchanged.
y = (x == 0) ? 0 : y;
```

Example adjusted from literature, see references.

Example: Memory Model On Intel 80x86 Processors

Example: Effects Of Intel 80x86 Memory Ordering Model

```
A       dd      0
B       dd      0
```

Executed on one processor:

```
        mov     [A], 1
        mov     eax, [B]
```

Executed on another processor:

```
        mov     [B], 1
        mov     eax, [A]
```

It is possible for both processors to finish with EAX containing 0.

Example: Memory Model In Java

Example: Causality Loops With Happens-Before Consistency

```
int A = 0;
int B = 0;
```

Executed in one thread:

```
if (A == 1) B = 1;
```

Executed in another thread:

```
if (B == 1) A = 1;
```

Example adjusted from literature, see references.

Example: Effects Of Java Memory Ordering Model

```
int A = 0;
int B = 0;
```

Executed in one thread:

```
A = 1;
X = B;
```

Executed in another thread:

```
B = 1;
Y = A;
```

It is possible for threads to finish with both X and Y containing 0.

What Is The Interface

Atomic Operations

Windows Atomic Operations

```
LONG InterlockedIncrement (PLONG Addend);
LONG InterlockedDecrement (PLONG Addend);
LONG InterlockedExchange (LPLONG lplTarget, LONG lValue);

PSLIST_ENTRY InterlockedPushEntrySList (
  PSLIST_HEADER ListHead,
  PSLIST_ENTRY ListEntry);
PSLIST_ENTRY InterlockedPopEntrySList (
  PSLIST_HEADER ListHead);
```

GCC Atomic Operations

```
type __sync_fetch_and_{add,sub,or,and,xor,nand} (type  *ptr, type value, ...);
type __sync_{add,sub,or,and,xor,nand}_and_fetch (type  *ptr, type value, ...);

bool __sync_bool_compare_and_swap (type  *ptr, type oldval type newval, ...);
type __sync_val_compare_and_swap (type *ptr, type oldval type newval, ...);

type __sync_lock_test_and_set (type  *ptr, type value, ...);
void __sync_lock_release (type *ptr, ...)
```

- implemented for common scalars and pointers

- typically supported by processor instructions
- unsupported operations compiled as function calls

C++ Atomic Operations

```
template<typename T> struct atomic
{
  public:

    bool is_lock_free () { ... }

    T operator= (T i) { store (i); return (i); }
    void store (T i, memory_order m = memory_order_seq_cst) { ... }
    T load (memory_order m = memory_order_seq_cst) { ... }

    T operator++ () { ... }
    T operator-- () { ... }
    T fetch_add (T i, memory_order m = memory_order_seq_cst) { ... }
    T fetch_sub (T i, memory_order m = memory_order_seq_cst) { ... }

    T exchange (T i, memory_order m = memory_order_seq_cst) { ... }
    bool compare_exchange_weak (T& e, T i, memory_order succ, memory_order fail) { ...
    bool compare_exchange_strong (T& e, T i, memory_order succ, memory_order fail) { ..

    ...
}

enum memory_order
{
  memory_order_relaxed, // No ordering constraints
  memory_order_consume, // Load will be consume operation (no reordering of dependent o
  memory_order_acquire, // Load will be acquire operation (no reordering of arbitrary o
  memory_order_release, // Store will be release operation
  memory_order_acq_rel, // Load-modify-store will be acquire and release operation
  memory_order_seq_cst  // Any operation will be totally ordered acquire and release op
};
```

Barriers

GCC Barrier Constructs

```
asm ("" : : : "memory");
```

- empty assembler statement
- no output and input operands
- indicates memory accessed in undefined manner
- no output operands imply important side effects
- does not include processor barrier operation

```
// Full hardware memory barrier
__sync_synchronize (...);
```

Visual C++ Barrier Constructs

```
void _ReadBarrier (void);
void _WriteBarrier (void);
void _ReadWriteBarrier (void);
```

- compiler intrinsic functions
- do not impact variables compiler knows are local
- different behavior with different compiler versions
- does not include processor barrier operation

```
// Full hardware memory barrier
void MemoryBarrier (void);
```

Posix Barrier Interface

```
int pthread_barrier_init (
  pthread_barrier_t *barrier,
  const pthread_barrierattr_t *attr,
  unsigned count);
int pthread_barrier_destroy (
  pthread_barrier_t *barrier);

int pthread_barrier_wait (
  pthread_barrier_t *barrier);
```

Locks

Posix Mutex Interface

```
int pthread_mutex_init (pthread_mutex_t *mutex,
                        const pthread_mutex_attr_t *mutexattr);
int pthread_mutex_destroy (pthread_mutex_t *mutex);

int pthread_mutex_lock (pthread_mutex_t *mutex);
int pthread_mutex_trylock (pthread_mutex_t *mutex);
int pthread_mutex_timedlock (pthread_mutex_t *restrict mutex,
                             const struct timespec *abs_timeout);
int pthread_mutex_unlock (pthread_mutex_t *mutex);
```

- error checking mutex
- recursive mutex
- fast mutex

Posix Spin Lock Interface

```
int pthread_spin_init (pthread_spinlock_t *lock, int pshared);
int pthread_spin_destroy (pthread_spinlock_t *lock);

int pthread_spin_lock (pthread_spinlock_t *lock);
int pthread_spin_trylock (pthread_spinlock_t *lock);
```

```
int pthread_spin_unlock (pthread_spinlock_t *lock);
```

- PTHREAD_PROCESS_PRIVATE - only for threads from the same process
- PTHREAD_PROCESS_SHARED - for any thread that can access the memory with the lock

Linux Futex Interface

```
int sys_futex (void *futex, int op, int val,
               const struct timespec *timeout)
```

- FUTEX_WAIT - verify val and sleep if unchanged
- FUTEX_WAKE - wake at most val processes
- FUTEX_CMP_REQUEUE - wake at most val processes and requeue rest

Trivial Mutex Using Futex

```
class mutex
{
  private:

    // Mutex state variable, zero means free.
    int val = 0;

  public:

    void lock ()
    {
      int old;

      // Atomically increment the state and
      // get the old value, which should be
      // zero if mutex was free.
      while ((old = atomic_inc (val)) != 0)
      {
        // The old value was not zero, meaning mutex was not free.
        // Wait unless the value has changed since the increment.
        futex_wait (&val, old + 1);
      }
    }

    void unlock ()
    {
      val = 0;
      // Wake a waiting caller if any.
      futex_wake (&val, 1);
    }
}
```

- unlock always calls kernel which is slow
- contention causes cache ping pong
- contention causes counter overflow

* *Source: Ulrich Drepper*

Windows Critical Section Interface

```
void InitializeCriticalSection (LPCRITICAL_SECTION lpCriticalSection);
BOOL InitializeCriticalSectionAndSpinCount (
  LPCRITICAL_SECTION lpCriticalSection,
  DWORD dwSpinCount);

void EnterCriticalSection (LPCRITICAL_SECTION lpCriticalSection);
BOOL TryEnterCriticalSection (LPCRITICAL_SECTION lpCriticalSection);
void LeaveCriticalSection (LPCRITICAL_SECTION lpCriticalSection);
```

- dwSpinCount - spin count to avoid context switch on multiple processors

Windows Mutex Interface

```
HANDLE CreateMutex (LPSECURITY_ATTRIBUTES lpsa,
                    BOOL fInitialOwner,
                    LPTSTR lpszMutexName);
HANDLE OpenMutex (DWORD dwDesiredAccess,
                  BOOL bInheritHandle,
                  LPCTSTR lpName);

DWORD WaitForSingleObject (
  HANDLE hHandle,
  DWORD dwMilliseconds);

BOOL ReleaseMutex (HANDLE hMutex);
```

Java Lock Support Interface

```
class java.util.concurrent.locks.LockSupport {
  static void park<8203>();
  static void parkNanos<8203>(long nanos);
  static void parkUntil<8203>(long deadline);

  static void unpark (Thread thread);

  ...
}
```

- thread may possess parking permit
- parking blocks thread until permit becomes available
- unparking provides permit even when thread not parked

Read Write Locks

Posix Read Write Lock Interface

```
int pthread_rwlock_init (pthread_rwlock_t *rwlock,
                         const pthread_rwlockattr_t *attr);
int pthread_rwlock_destroy (pthread_rwlock_t *rwlock);
```

```
int pthread_rwlock_rdlock (pthread_rwlock_t *rwlock);
int pthread_rwlock_wrlock (pthread_rwlock_t *rwlock);

int pthread_rwlock_tryrdlock (pthread_rwlock_t *rwlock);
int pthread_rwlock_trywrlock (pthread_rwlock_t *rwlock);

int pthread_rwlock_unlock (pthread_rwlock_t *rwlock);
```

Windows Slim Reader Writer Interface

```
VOID InitializeSRWLock (PSRWLOCK SRWLock);

VOID AcquireSRWLockShared (PSRWLOCK SRWLock);
VOID AcquireSRWLockExclusive (PSRWLOCK SRWLock);

VOID ReleaseSRWLockShared (PSRWLOCK SRWLock);
VOID ReleaseSRWLockExclusive (PSRWLOCK SRWLock);
```

Seq Lock

Linux Seq Lock Interface

```
seqlock_init (seqlock_t *sl);

void write_seqlock (seqlock_t *sl);
void write_sequnlock (seqlock_t *sl);
int write_tryseqlock (seqlock_t *sl);

read_seqbegin (const seqlock_t *sl);
int read_seqretry (const seqlock_t *sl, unsigned start);
```

- version incremented once before write and once after write
- readers require same version before read and after read
- even version means consistent, odd version means retry

Seq Lock Usage Example

```
do
{
  start = read_seqbegin (&sl);
  ...
}
while (read_seqretry (&sl, start);
```

Read Copy Update

Linux Read Copy Update Interface

```
void rcu_read_lock ();
void rcu_read_unlock ();
```

```
typeof(ptr) rcu_assign_pointer (ptr, val);
typeof(ptr) rcu_dereference (ptr);

void synchronize_rcu ();
```

- readers access last consistent version
- new version hidden while inconsistent
- writer atomically installs new version
- writer releases old version when no old version reader exists

Semaphores

Unix Semaphore Interface

```
int semget (key_t key, int nsems, int semflg);
```

- IPC_PRIVATE - private key value
- IPC_CREAT - object with key can be created
- IPC_EXCL - object with key must not exist

```
int semop (int semid, struct sembuf *sops, unsigned nsops);
int semtimedop (int  semid, struct sembuf *sops, unsigned nsops, struct timespec *timec
```

- positive - add value to semaphore
- zero - wait for zero value of semaphore
- negative - subtract value from semaphore or wait

```
key_t ftok (const char *pathname, int proj_id);
```

Posix Semaphore Interface

```
int sem_init (sem_t *sem, int pshared, unsigned int value);
int sem_destroy (sem_t *sem);

sem_t *sem_open (const char *name, int oflag,
                 mode_t mode, unsigned int value);
int sem_unlink (const char *name);

int sem_wait (sem_t *sem);
int sem_trywait (sem_t *sem);
int sem_timedwait (sem_t *restrict sem,
                   const struct timespec *abs_timeout);

int sem_post (sem_t *sem);

int sem_getvalue (sem_t *sem, int *sval);
```

- named - semaphore for synchronization between processes
- unnamed - semaphore for synchronization within a process

Windows Semaphore Interface

```
HANDLE CreateSemaphore (LPSECURITY_ATTRIBUTE lpsa,
                        LONG cSemInitial,
                        LONG cSemMax,
                        LPTSTR lpszSemName);
HANDLE OpenSemaphore (DWORD dwDesiredAccess,
                      BOOL bInheritHandle,
                      LPCTSTR lpName);

DWORD WaitForSingleObject (
  HANDLE hHandle,
  DWORD dwMilliseconds);

BOOL ReleaseSemaphore (HANDLE hSemaphore,
                       LONG cRelease,
                       LPLONG lplPrevious);
```

Condition Variables

Posix Condition Variable Interface

```
int pthread_cond_init (pthread_cond_t *cond, pthread_condattr_t *cond_attr);
int pthread_cond_destroy (pthread_cond_t *cond);

int pthread_cond_signal (pthread_cond_t *cond);
int pthread_cond_broadcast (pthread_cond_t *cond);
int pthread_cond_wait (pthread_cond_t *cond, pthread_mutex_t *mutex);
int pthread_cond_timedwait (pthread_cond_t *cond,
                            pthread_mutex_t *mutex,
                            const struct timespec *abstime);
```

Windows Condition Variable Interface

```
VOID InitializeConditionVariable (
  PCONDITION_VARIABLE ConditionVariable);

BOOL SleepConditionVariableCS (
  PCONDITION_VARIABLE ConditionVariable,
  PCRITICAL_SECTION CriticalSection,
  DWORD dwMilliseconds);
BOOL SleepConditionVariableSRW (
  PCONDITION_VARIABLE ConditionVariable,
  PSRWLOCK SRWLock,
  DWORD dwMilliseconds,
  ULONG Flags);

VOID WakeConditionVariable (
  PCONDITION_VARIABLE ConditionVariable);
VOID WakeAllConditionVariable (
  PCONDITION_VARIABLE ConditionVariable);
```

Events

Windows Event Interface

```
HANDLE CreateEvent (LPSECURITY_ATTRIBUTES lpsa,
                    BOOL fManualReset,
                    BOOL fInitialState,
                    LPTSTR lpszEventName);
HANDLE OpenEvent (DWORD dwDesiredAccess,
                  BOOL bInheritHandle,
                  LPCTSTR lpName);
```

- manual reset - remains signalled after set until reset
- automatic reset - returns to unsignalled after thread wake up

```
BOOL SetEvent (HANDLE hEvent);
BOOL ResetEvent (HANDLE hEvent);
BOOL PulseEvent (HANDLE hEvent);
```

Monitors

Monitors

```
class MyClass {
  int a;
  synchronized Foo () {
    a = 1;
  }
  synchronized Bar () {
    a = 2;
  }
}

MyClass b = new MyClass ();
synchronized (b) {
  ...
}
```

Guards

Guards

```
task body Foo is
i,j : integer;
begin
...
select
when j > 0 =>
accept Xyzzy (n : integer) do
i := n;
end Xyzzy;
or
...
end select;
```

```
...
end Foo;

task body Bar is
begin
Xyzzy (1);
end Bar;
```

- execute rendez vous if possible
- execute conditional branch if possible
- throw exception if neither of the two is possible

Chapter 3. Memory Management

Management Among Processes

Separating Multiple Processes

Software Implementation

Example: Solaris

Pager Operations

- advise - access optimization hints
- checkprot - check whether access is allowed
- fault - handle page fault
- lockop - lock or unlock a page
- swapout - swap out maximum number of pages
- sync - write out dirty pages

What Is The Interface

MMap And MUnmap System Calls

```
void *mmap (
  void *start,
  size_t length,
  int prot,
  int flags,
  int fd,
  off_t offset);

int munmap (
  void *start,
  size_t length);
```

- MAP_FIXED - require supplied address
- MAP_SHARED - share mapping with other processes
- MAP_PRIVATE - create a copy on write mapping
- MAP_ANONYMOUS - create mapping backed in swap

- MAP_GROWSDOWN - block grows down rather than up
- MAP_POPULATE - fetch pages into memory
- MAP_HUGETLB - allocate large pages
- MAP_LOCKED - lock pages in memory

MAdvise System Call

```
int madvise (
  void *addr,
  size_t length,
  int advice);
```

- MADV_NORMAL - default treatment
- MADV_RANDOM - accessed in random order (read ahead less useful)
- MADV_SEQUENTIAL - accessed in sequential order (read ahead more useful, good victim after access)
- MADV_WILLNEED - access in near future likely (read ahead useful now)
- MADV_DONTNEED - access in near future unlikely (good victim now)
- MADV_HUGEPAGE - merge pages into huge page where possible
- MADV_MERGEABLE - merge pages with identical content

MBind System Call

```
int set_mempolicy (
  int mode,
  unsigned long *nodemask, unsigned long maxnode);
int get_mempolicy (
  int *mode,
  unsigned long *nodemask, unsigned long maxnode,
  unsigned long addr,
  unsigned long flags);

int mbind (
  void *addr, unsigned long len,
  int mode,
  unsigned long *nodemask, unsigned long maxnode,
  unsigned flags);
```

- MPOL_DEFAULT - allocate on node that requests memory
- MPOL_BIND - allocate from listed nodes, exhaust node before using next
- MPOL_PREFERRED - allocate as close to first listed node as possible
- MPOL_INTERLEAVE - allocate from listed nodes, cycle across nodes

- MPOL_MF_MOVE - move already allocated pages to satisfy policy
- MPOL_MF_STRICT - fail if already allocated pages do not satisfy policy

VirtualAlloc System Call

```
LPVOID VirtualAlloc (
  LPVOID lpAddress,
  SIZE_T dwSize,
  DWORD flAllocationType,
  DWORD flProtect);
```

- MEM_RESET - drop data currently in block
- MEM_COMMIT - reserve address range and storage for block
- MEM_RESERVE - reserve address range but not storage for block

- MEM_TOP_DOWN - allocate with as high address as possible
- MEM_WRITE_WATCH - keep list of pages that were modified

GetWriteWatch System Call

```
UINT GetWriteWatch (
  DWORD dwFlags,
  PVOID lpBaseAddress,
  SIZE_T dwRegionSize,
  PVOID *lpAddresses,
  PULONG_PTR lpdwCount,
  PULONG lpdwGranularity);
```

- WRITE_WATCH_FLAG_RESET - stop watching

Allocation Within A Process

Process Memory Layout

Example: Virtual Address Space Of A Linux Process

Process Address Space Layout

```
> cat /proc/self/maps
00111000-00234000 r-xp 00000000 03:01 3653725    /lib/libc-2.3.5.so
00234000-00236000 r-xp 00123000 03:01 3653725    /lib/libc-2.3.5.so
00236000-00238000 rwxp 00125000 03:01 3653725    /lib/libc-2.3.5.so
00238000-0023a000 rwxp 00238000 00:00 0
007b5000-007cf000 r-xp 00000000 03:01 3653658    /lib/ld-2.3.5.so
007cf000-007d0000 r-xp 00019000 03:01 3653658    /lib/ld-2.3.5.so
007d0000-007d1000 rwxp 0001a000 03:01 3653658    /lib/ld-2.3.5.so
008ed000-008ee000 r-xp 008ed000 00:00 0          [vdso]
08048000-0804d000 r-xp 00000000 03:01 3473470    /bin/cat
0804d000-0804e000 rw-p 00004000 03:01 3473470    /bin/cat
09ab8000-09ad9000 rw-p 09ab8000 00:00 0          [heap]
b7d88000-b7f88000 r--p 00000000 03:01 6750409    /usr/lib/locale/locale-archive
b7f88000-b7f89000 rw-p b7f88000 00:00 0
b7f96000-b7f97000 rw-p b7f96000 00:00 0
bfd81000-bfd97000 rw-p bfd81000 00:00 0          [stack]
```

Stack

Example: Stack On Intel 80x86 Processors

Stack Allocation And Access

```
void SomeProcedure (int anArgument)
{
  int aVariable;
  aVariable = anArgument;
}

SomeProcedure:

    push    ebp             ;save original value of EBP on stack
    mov     ebp,esp         ;store top of stack address in EBP
    sub     esp,4           ;allocate space for aVariable on stack

    mov     eax,[ebp+8]     ;fetch anArgument into EAX, which is
                            ;8 bytes below the stored top of stack
    mov     [ebp-4],eax     ;store EAX into aVariable, which is
                            ;4 bytes above the stored top of stack

    mov     esp,ebp         ;free space allocated for aVariable
    pop     ebp             ;restore original value of EBP
    ret                     ;return to the caller
```

Heap

Heap Allocators

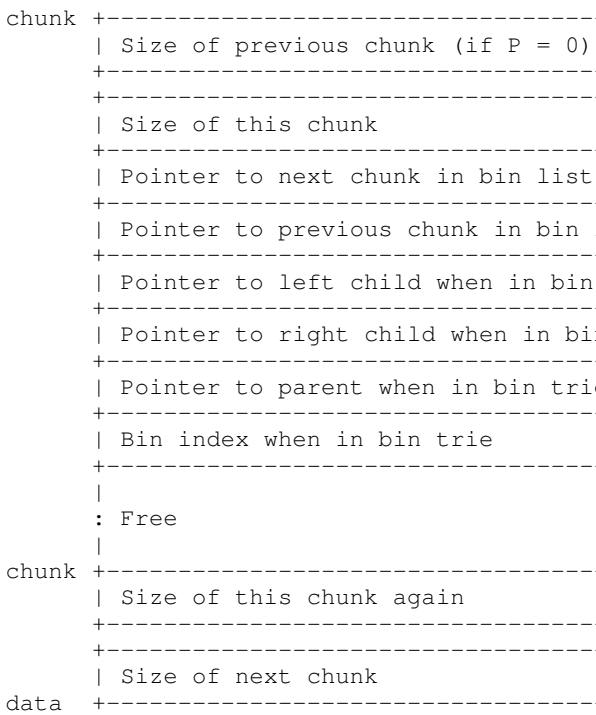
Example: dlmalloc Heap Allocator

Allocated Chunk Structure

```
chunk +-----------------------------------------------+
      | Size of previous chunk (if P = 0)             |
      +-----------------------------------------+---+
      +-------------------------------------+---+ | P |
      | Size of this chunk                  | 1 | +---+
data  +-------------------------------------+---+-----+
      |                                               |
      : User data (size - sizeof (size_t) bytes)      :
      |                                               |
chunk +-----------------------------------------------+
      | Size of this chunk again                      |
      +-----------------------------------------+---+
      +-------------------------------------+---+ | 1 |
      | Size of next chunk                  | U | +---+
data  +-------------------------------------+---+-----+

// Adjusted from dlmalloc source code comments.
```

Free Chunk Structure




```
// Adjusted from dlmalloc source code comments.
```

Example: Posix Heap Allocator Interface

Posix Heap Allocator Interface

```
void *malloc (size_t size);
void *calloc (size_t nmemb, size_t size);
void *realloc (void *ptr, size_t size);
void free (void *ptr);

// Aligned allocation for power-of-two alignment
int posix_memalign (void **memptr, size_t alignment, size_t size);

// Aligned allocation for power-of-two alignment (deprecated)
void *memalign (size_t boundary, size_t size);
// Aligned allocation for page-size alignment (deprecated)
void *valloc (size_t size);
```

Linux Heap Allocator Interface

```
int mallopt (int param, int value);
```

- M_CHECK_ACTION - control error reporting
- M_PERTURB - pattern fill on alloc and free

```
struct mallinfo mallinfo (void);
```

```
struct mallinfo {
  int arena;     // Allocated space except mapped regions
  int ordblks;   // Number of free blocks except fast bins
  int smblks;    // Number of free fast bins
  int hblks;     // Number of mapped regions
  int hblkhd;    // Space in mapped regions
  int usmblks;   // Maximum space allocated so far (only when single thread)
  int fsmblks;   // Space in free fast bins
  int uordblks;  // Total allocated space
  int fordblks;  // Total free space
  int keepcost;  // Space at heap end that can be released
};
```

Example: Linux Kernel Slab Allocator

Slab Cache System Calls

```
// Create a slab cache
kmem_cache_t * kmem_cache_create (
  const char *name,
  size_t size,
  size_t align,
  unsigned long flags,
  void (* ctor) (void *, kmem_cache_t *, unsigned long),
  void (* dtor) (void *, kmem_cache_t *, unsigned long));
```

- name - slab cache name for debugging
- size - object size for this slab cache
- align - optional object alignment
- ctor - optional object constructor
- dtor - optional object destructor

```
// Allocate and free slabs of the cache
void *kmem_cache_alloc (kmem_cache_t *cachep, int flags);
void kmem_cache_free (kmem_cache_t *cachep, void *objp);
```

- SLAB_DEBUG_FREE - check correctness of free
- SLAB_RED_ZONE - create red zone after objects
- SLAB_POISON - fill free objects with poison data
- SLAB_STORE_USER - remember return address to last object user
- SLAB_CACHE_DMA - allocate memory for use with DMA
- SLAB_HWCACHE_ALIGN - align objects on cache lines

Slab Allocator Usage Statistics

```
> cat /proc/slabinfo
slabinfo - version: 2.1
# name            <active_objs> <num_objs> <objsize> <objperslab> <pagesperslab> ...
ext4_inode_cache         49392      49392      1016           16              4
ext4_free_data             128        128        64           64              1
ext4_xattr                  92         92        88           46              1
blkdev_requests            273        273       376           21              2
blkdev_queue                30         30      2080           15              8
vm_area_struct            3520       3520       184           22              1
task_struct                160        160      1952           16              8
inode_cache              11899      11928       568           14              2
dentry                  401373     401373       192           21              1
...
```

Chapter 4. Device Management

Device Drivers

Asynchronous Requests

Example: Linux Tasklets

Soft IRQ Example

```
// Registers softirq handler
extern void open_softirq (
  int nr,
  void (*action)(struct softirq_action*),
  void *data);

void open_softirq (...)
{
  softirq_vec [nr].data = data;
  softirq_vec [nr].action = action;
}

// Schedules softirq handler
inline fastcall void raise_softirq_irqoff (unsigned int nr)
{
  or_softirq_pending (1UL &lt;&lt; (nr));
  if (!in_interrupt ()) wakeup_softirqd ();
}
```

- Multiprocessor system can execute multiple soft irqs concurrently.
- One soft irq can execute concurrently on multiple processors.

Tasklet Interface Example

```
#define DECLARE_TASKLET(name, func, data) \
struct tasklet_struct name = { NULL, 0, ATOMIC_INIT(0), func, data }

// Schedule tasklet for execution on current processor
void tasklet_schedule (struct tasklet_struct *t);

void tasklet_disable (struct tasklet_struct *t);
void tasklet_enable (struct tasklet_struct *t);
```

- Scheduling guarantees the tasklet will be called at least once.
- Multiprocessor system can execute multiple tasklets concurrently.
- One tasklet never executes concurrently on multiple processors.

Tasklet Handling Example

```
static void tasklet_action (struct softirq_action *a)
{
  struct tasklet_struct *list;

  // Get the entire tasklet queue
  local_irq_disable ();
    list = __get_cpu_var (tasklet_vec).head;
    __get_cpu_var (tasklet_vec).head = NULL;
    __get_cpu_var (tasklet_vec).tail = &__get_cpu_var (tasklet_vec).head;
  local_irq_enable ();

  // Go through the queue tasklet by tasklet
  while (list) {
    struct tasklet_struct *t = list;

    list = list->next;

    // Necessary synchronization with other processors
    if (tasklet_trylock (t)) {
      if (!atomic_read (&t->count)) {
        if (!test_and_clear_bit (TASKLET_STATE_SCHED, &t->state))
          BUG ();
        t->func (t->data);
        tasklet_unlock (t);
        continue;
      }
      tasklet_unlock (t);
    }

    // Put the executed tasklets back into queue
    local_irq_disable();
      t->next = NULL;
      *__get_cpu_var(tasklet_vec).tail = t;
      __get_cpu_var(tasklet_vec).tail = &(t->next);
      __raise_softirq_irqoff(TASKLET_SOFTIRQ);
    local_irq_enable();
  }
}
```

Work Queue Example

```
#define DECLARE_WORK(name, func, data) \
struct work_struct name = { data, NULL, func)

// Create a work queue with a kernel thread to serve it
struct workqueue_struct *create_workqueue (const char *name);

// Request executing work by a given work queue
int queue_work (
  struct workqueue_struct *queue,
  struct work_struct *work);
int queue_delayed_work (
  struct workqueue_struct *queue,
  struct work_struct *work,
  unsigned long delay);

// Request executing work by the default work queue
int schedule_work (
  struct work_struct *work);
int schedule_delayed_work (
  struct work_struct *work,
  unsigned long delay);
```

```
void flush_workqueue (struct workqueue_struct *queue);
```

Example: Windows Deferred Procedure Calls

DPC Example

```
// Registers DPC for a device
VOID IoInitializeDpcRequest (
  IN PDEVICE_OBJECT DeviceObject,
  IN PIO_DPC_ROUTINE DpcRoutine
);

// Schedules DPC for a device
VOID IoRequestDpc (
  IN PDEVICE_OBJECT DeviceObject,
  IN PIRP Irp,
  IN PVOID Context
);

// DPC
VOID DpcForIsr (
  IN PKDPC Dpc,
  IN struct _DEVICE_OBJECT *DeviceObject,
  IN struct _IRP *Irp,
  IN PVOID Context
);
```

- DeviceObject - device instance
- DpcRoutine - deferred procedure call routine
- Irp - structure describing the request being processed
- Context - driver context to be passed to the routine

Synchronous Requests

Example: Linux Driver Model

Busses In Linux Driver Model

```
> ls -R /sys/bus
/sys/bus:
pci  pci_express  pcmcia  scsi  usb
/sys/bus/pci:
devices  drivers
/sys/bus/pci/devices:
0000:00:00.0  0000:00:1a.7  0000:00:1c.3  0000:00:1d.7  0000:00:1f.3
0000:00:01.0  0000:00:1b.0  0000:00:1c.4  0000:00:1e.0  0000:01:00.0
/sys/bus/pci/drivers:
agpgart-intel  ata_piix  ehci_hcd  ohci_hcd  uhci_hcd  ahci  e1000  HDA Intel
...
```

Devices In Linux Driver Model

```
> ls -R /sys/devices
/sys/devices:
pci0000:00
/sys/devices/pci0000:00:
0000:00:19.0
/sys/devices/pci0000:00/0000:00:19.0:
class  config  device  driver  irq  net  power  vendor
/sys/devices/pci0000:00/0000:00:19.0/net:
eth0
/sys/devices/pci0000:00/0000:00:19.0/net/eth0:
address  broadcast  carrier  device  features  flags  mtu  power  statistics
...
```

Device Insertion Notification In Linux Driver Model

```
> udevmonitor --env
UEVENT[12345.67890] add /devices/pci0000:00/0000:00:1a.7/usb1/1-3/1-3:1.0 (usb)
ACTION=add
DEVPATH=/devices/pci0000:00/0000:00:1a.7/usb1/1-3/1-3:1.0
SUBSYSTEM=usb
DEVTYPE=usb_interface
DEVICE=/proc/bus/usb/001/006
PRODUCT=457/151/100
INTERFACE=8/6/80
MODALIAS=usb:v0457p0151d0100dc00dsc00dp00ic08isc06ip50
```

Devices

Busses

Example: SCSI

SCSI Inquiry Command

Bit / Byte	7	6	5	4	3	2	1	0
0	Operation Code: Inquiry (12h)							
1	Logical Unit Number				Reserved			EVPD
2	Page Code							
3	Reserved							
4	Allocation Length: Inquiry Reply Length (96)							
5	Control							

SCSI Inquiry Response

Bit / Byte	7	6	5	4	3	2	1	0
0	Peripheral Qualifier			Peripheral Device Type				
1	RMB	Device-Type Modifier						
2	ISO Version		ECMA Version			ANSI Version		
3	AENC	TrmIOP	Reserved			Response Data Format		
4	Additional Length (n-4)							
5	Reserved							
6	Reserved							
7	RelAdr	WBus32	WBus16	Sync	Linked	Reserved	CmdQue	SftRe
8	(MSB)							
15				Vendor Identification				(LSB)
16	(MSB)							
31				Product Identification				(LSB)
32	(MSB)							
35				Product Revision Level				(LSB)
36								
55				Vendor Specific				
56								
95				Reserved				
96								
n				Additional Vendor Specific				

SCSI Device Listing

```
> cat /proc/scsi/scsi
Attached devices:
Host: scsi0 Channel: 00 Id: 00 Lun: 00
  Vendor: QUANTUM  Model: ATLAS10K2-TY184L Rev: DA40
  Type:   Direct-Access                    ANSI SCSI revision: 03
Host: scsi1 Channel: 00 Id: 05 Lun: 00
  Vendor: NEC      Model: CD-ROM DRIVE:466 Rev: 1.06
  Type:   CD-ROM                           ANSI SCSI revision: 02
Host: scsi2 Channel: 00 Id: 00 Lun: 00
  Vendor: PLEXTOR  Model: DVDR   PX-708A   Rev: 1.02
  Type:   CD-ROM                           ANSI SCSI revision: 02
```

Example: PCI

PCI Device Listing

```
> lspci -t
-[0000:00]-+-00.0
           +-01.0-[0000:01]----00.0
           +-02.0-[0000:02-03]----1f.0-[0000:03]----00.0
           +-1e.0-[0000:04]--+-0b.0
           |                 +-0c.0
           |                 \-0d.0
           +-1f.0
           +-1f.1
           +-1f.2
           +-1f.3
           +-1f.4
           \-1f.5
```

PCI Brief Device Information

```
> lspci
00:00.0 Host bridge: Intel Corp. 82860 860 (Wombat) Chipset Host Bridge (MCH) (rev 04)
00:01.0 PCI bridge: Intel Corp. 82850 850 (Tehama) Chipset AGP Bridge (rev 04)
00:02.0 PCI bridge: Intel Corp. 82860 860 (Wombat) Chipset AGP Bridge (rev 04)
00:1e.0 PCI bridge: Intel Corp. 82801 PCI Bridge (rev 04)
00:1f.0 ISA bridge: Intel Corp. 82801BA ISA Bridge (LPC) (rev 04)
00:1f.1 IDE interface: Intel Corp. 82801BA IDE U100 (rev 04)
00:1f.2 USB Controller: Intel Corp. 82801BA/BAM USB (Hub #1) (rev 04)
00:1f.3 SMBus: Intel Corp. 82801BA/BAM SMBus (rev 04)
00:1f.4 USB Controller: Intel Corp. 82801BA/BAM USB (Hub #2) (rev 04)
00:1f.5 Multimedia audio controller: Intel Corp. 82801BA/BAM AC'97 Audio (rev 04)
01:00.0 VGA compatible controller: ATI Technologies Inc Radeon RV100 QY [Radeon 7000/VE
02:1f.0 PCI bridge: Intel Corp. 82806AA PCI64 Hub PCI Bridge (rev 03)
03:00.0 PIC: Intel Corp. 82806AA PCI64 Hub Advanced Programmable Interrupt Controller (
04:0b.0 Ethernet controller: 3Com Corporation 3c905C-TX/TX-M [Tornado] (rev 78)
04:0c.0 FireWire (IEEE 1394): Texas Instruments TSB12LV26 IEEE-1394 Controller (Link)
04:0d.0 Ethernet controller: Intel Corp. 82544EI Gigabit Ethernet Controller (Copper) (
```

PCI Detailed Device Information

```
> lspci -vvs 04:0b.0
04:0b.0 Ethernet controller: 3Com Corporation 3c905C-TX/TX-M [Tornado] (rev 78)
        Subsystem: Dell: Unknown device 00d8
        Control: I/O+ Mem+ BusMaster+ SpecCycle- MemWINV+ VGASnoop- ParErr- Stepping- S
        Status: Cap+ 66Mhz- UDF- FastB2B- ParErr- DEVSEL=medium >TAbort- <TAbort- <MAbo
        Latency: 64 (2500ns min, 2500ns max), Cache Line Size 10
        Interrupt: pin A routed to IRQ 23
        Region 0: I/O ports at dc80 [size=128]
        Region 1: Memory at ff3ffc00 (32-bit, non-prefetchable) [size=128]
        Expansion ROM at ff400000 [disabled] [size=128K]
        Capabilities: [dc] Power Management version 2
                Flags: PMEClk- DSI- D1+ D2+ AuxCurrent=0mA PME(D0+,D1+,D2+,D3hot+,D3col
                Status: D0 PME-Enable- DSel=0 DScale=2 PME-
```

Example: USB

USB Device Listing

```
> lsusb -t
Bus#  1
`-Dev#   1 Vendor 0x0000 Product 0x0000
  `-Dev#   2 Vendor 0x046d Product 0xc01b
```

USB Brief Device Information

```
> lsusb
Bus 001 Device 002: ID 046d:c01b Logitech, Inc. MX310 Optical Mouse
Bus 001 Device 001: ID 0000:0000
```

USB Detailed Device Information

```
> lsusb -vv -s 1:2

Bus 001 Device 002: ID 046d:c01b Logitech, Inc. MX310 Optical Mouse
Device Descriptor:
  bLength                18
  bDescriptorType         1
  bcdUSB               2.00
  bDeviceClass            0 (Defined at Interface level)
  bDeviceSubClass         0
  bDeviceProtocol         0
  bMaxPacketSize0         8
  idVendor           0x046d Logitech, Inc.
  idProduct          0xc01b MX310 Optical Mouse
  bcdDevice           18.00
  iManufacturer           1 Logitech
  iProduct                2 USB-PS/2 Optical Mouse
  iSerial                 0
  bNumConfigurations      1
  Configuration Descriptor:
    bLength                 9
    bDescriptorType         2
    wTotalLength           34
    bNumInterfaces          1
    bConfigurationValue     1
    iConfiguration          0
    bmAttributes         0xa0
      Remote Wakeup
    MaxPower               98mA
    Interface Descriptor:
      bLength                 9
      bDescriptorType         4
      bInterfaceNumber        0
      bAlternateSetting       0
      bNumEndpoints           1
      bInterfaceClass         3 Human Interface Devices
      bInterfaceSubClass      1 Boot Interface Subclass
      bInterfaceProtocol      2 Mouse
      iInterface              0
      Endpoint Descriptor:
        bLength                 7
        bDescriptorType         5
        bEndpointAddress     0x81  EP 1 IN
        bmAttributes            3
          Transfer Type            Interrupt
```

```
  Synch Type               none
  Usage Type               Data
wMaxPacketSize     0x0005  bytes 5 once
bInterval             10
```

Disk Storage Devices

Partitioning

Example: IBM Volume Partitioning

IBM Partition Table

```
Offset  Length  Contents
------------------------------------------------------------------------------
000h 446 Boot loader code
1BEh 16 Partition 1
1CEh 16 Partition 2
1DEh 16 Partition 3
1EEh 16 Partition 4
1FEh 2 Magic (0AA55h)

Offset  Length  Contents
------------------------------------------------------------------------------
00h 1 Bootable flag
01h 1 Starting head number
02h 2 Starting sector and track
04h 1 System ID
05h 1 Ending head number
06h 2 Ending sector and track
08h 4 Starting sequential sector number
0Ch 4 Length in sequential sectors
```

Example: GPT Volume Partitioning

GPT Partition Table

```
Offset  Length  Contents
------------------------------------------------------------------------------
000h    8       Magic ("EFI PART")
008h    4       Version
00Ch    4       Header size (typically 05Ch)
010h    4       Header CRC32
014h    4       0
018h    8       This header LBA address
020h    8       Backup header LBA address
028h    8       First data block LBA address
030h    8       Last data block LBA address
038h    16      Disk UUID
048h    8       Partition array LBA address (typically 2)
050h    4       Partition array entry count
054h    4       Partition array entry size (typically 128)
058h    4       Partition array CRC32

Offset  Length  Contents
```

```
------------------------------------------------------------------------------
000h    16      Partition type UUID
010h    16      Partition UUID
020h    8       First data block LBA address
028h    8       Last data block LBA address
030h    8       Flags
038h    72      Partition name (UTF16)
```

Example: Linux Logical Volume Management

Volume Group Example

```
> vgdisplay

  --- Volume group ---
  VG Name               volumes
  System ID
  Format                lvm2
  Metadata Areas        2
  Metadata Sequence No  10
  VG Access             read/write
  VG Status             resizable
  MAX LV                0
  Cur LV                3
  Open LV               3
  Max PV                0
  Cur PV                2
  Act PV                2
  VG Size               1.27 TiB
  PE Size               32.00 MiB
  Total PE              41695
  Alloc PE / Size       24692 / 771.62 GiB
  Free  PE / Size       17003 / 531.34 GiB
  VG UUID               fbvtrb-GFbS-Nvf4-Ogg3-J4fX-dj83-ebh39q
```

Physical Volume Example

```
> pvdisplay --map

  --- Physical volume ---
  PV Name               /dev/md0
  VG Name               volumes
  PV Size               931.39 GiB / not usable 12.56 MiB
  Allocatable           yes
  PE Size               32.00 MiB
  Total PE              29804
  Free PE               17003
  Allocated PE          12801
  PV UUID               hvfcSD-FvSp-xJn4-lsR3-40Kx-LdDD-wvfGfV

  --- Physical Segments ---
  Physical extent 0 to 6875:
    Logical volume /dev/volumes/home
    Logical extents 0 to 6875
  Physical extent 6876 to 6876:
    Logical volume /dev/volumes/var
    Logical extents 11251 to 11251
  Physical extent 6877 to 12800:
    Logical volume /dev/volumes/home
    Logical extents 6876 to 12799
```

```
  Physical extent 12801 to 29803:
    FREE
```

Logical Volume Example

```
> lvdisplay --map

  --- Logical volume ---
  LV Name                /dev/volumes/home
  VG Name                volumes
  LV UUID                OAdf3v-zfI1-w5vq-tFVr-Sfgv-yvre-GWFb3v
  LV Write Access        read/write
  LV Status              available
  LV Size                400.00 GiB
  Current LE             12800
  Segments               2
  Allocation             inherit
  Read ahead sectors     auto
  - currently set to     256
  Block device           253:2

  --- Segments ---
  Logical extent 0 to 6875:
    Type  linear
    Physical volume /dev/md0
    Physical extents 0 to 6875

  Logical extent 6876 to 12799:
    Type  linear
    Physical volume /dev/md0
    Physical extents 6877 to 12800
```

Logical Volume Configuration

contiguous allocation
: allocate adjacent extents

arbitrary allocation
: allocate anywhere at all

normal allocation
: allocate using common sense

cling allocation
: allocate on the same physical volume

linear mapping
: a range of physical extents maps to a range of logical extents in linear order

striped mapping
: physical extents from multiple physical volumes map to a range of logical extents in cyclic order

snapshot volume
: a logical volume that is a copy-on-write image of another logical volume

mirror volume
: a logical volume that is a mirror image of another logical volume

sparse volume
: a logical volume whose storage is mapped upon write

Chapter 5. File Subsystem

File Subsystem

Poskytované abstrakce

- adresáře
- soubory

Základní požadavky

- ukládání velkého počtu i objemu dat
- co nejmenší kapacitní a časová režie
- odolnost proti výpadku systému
- zabezpečení proti neoprávněnému přístupu
- koordinace sdílení dat

Abstractions And Operations

Stream File Operations

Example: Linux Stream File Operations

Open And Close System Calls

```
int open (
  char *pathname,
  int flags);
int open (
  char *pathname,
  int flags, mode_t mode);

int creat (
  char *pathname,
  mode_t mode);

int close (int fd);
```

- O_RDONLY, O_WRONLY, O_RDWR
- O_CREAT, O_EXCL, O_TRUNC, O_APPEND
- O_NONBLOCK, O_SYNC

Seek System Call

```
off_t lseek (
  int fildes, off_t offset, int whence);
```

- SEEK_SET, SEEK_CUR, SEEK_END

Synchronous Read And Write System Calls

```
ssize_t read (int fd,
  void *buf, size_t count);
ssize_t write (int fd,
  void *buf, size_t count);

// File access that ignores seek position
ssize_t pread (int fd,
  void *buf, size_t count,
  off_t offset);
ssize_t pwrite (int fd,
  void *buf, size_t count,
  off_t offset);

// File access with scatter and gather
ssize_t readv (int fd,
  struct iovec *vector, int count);
ssize_t writev (int fd,
  struct iovec *vector, int count);

struct iovec {
  void    *iov_base;
  size_t  iov_len;
};
```

Asynchronous Read And Write System Calls

```
// Asynchronous single operation
int aio_read (struct aiocb *aiocbp);
int aio_write (struct aiocb *aiocbp);

// Query asynchronous operation status
int aio_error (struct aiocb *aiocbp);
// Get return status of completed operation
ssize_t aio_return (struct aiocb *aiocbp);

// Wait for completion of any listed operation
int aio_suspend (
  struct aiocb *cblist [],
  int n, struct timespec *timeout);

int aio_cancel (int fd, struct aiocb *aiocbp);

// Submit multiple asynchronous operations
int lio_listio (
  int mode, struct aiocb *list [],
  int nent, struct sigevent *sig);

struct aiocb {
  int              aio_fildes;
  off_t            aio_offset;
  void             *aio_buf;
  size_t           aio_nbytes;
  int              aio_reqprio;
  struct sigevent  aio_sigevent;
  int              aio_lio_opcode;
  ...
}
```

Advise System Calls

```
int posix_fadvise (
  int fd,
  off_t offset, off_t len,
  int advice);

int posix_fallocate (
  int fd,
  off_t offset, off_t len);
```

- POSIX_FADV_NORMAL - no advice
- POSIX_FADV_SEQUENTIAL - sequential access
- POSIX_FADV_RANDOM - random access
- POSIX_FADV_NOREUSE - data used once
- POSIX_FADV_WILLNEED - data will be used soon
- POSIX_FADV_DONTNEED - data will not be used soon

Example: Windows Stream File Operations

Open And Close System Calls

```
HFILE OpenFile (
  LPCSTR lpFileName,
  LPOFSTRUCT lpReOpenBuff,
  UINT uStyle);

HANDLE CreateFile (
  LPCTSTR lpFileName,
  DWORD dwDesiredAccess,
  DWORD dwShareMode,
  LPSECURITY_ATTRIBUTES lpSecurityAttributes,
  DWORD dwCreationDisposition,
  DWORD dwFlagsAndAttributes,
  HANDLE hTemplateFile);

HANDLE ReOpenFile (
  HANDLE hOriginalFile,
  DWORD dwDesiredAccess,
  DWORD dwShareMode,
  DWORD dwFlags);

BOOL CloseHandle (HANDLE hObject);
```

- OF_CREATE - create or truncate file
- OF_EXIST - open and close, used to test existence
- OF_PARSE - only fill the reopen structure
- OF_PROMPT - open a retry dialog if the file does not exist
- OF_REOPEN - use the reopen structure
- OF_VERIFY - compare timestamp with reopen structure

- FILE_SHARE_READ - allow concurrent reads
- FILE_SHARED_WRITE - allow concurrent writes
- FILE_SHARED_DELETE - allow concurrent deletion

- FILE_FLAG_DELETE_ON_CLOSE - used for temporary files
- FILE_FLAG_NO_BUFFERING - bypass memory manager, access must be sector aligned
- FILE_FLAG_OVERLAPPED - allow asynchronous operation
- FILE_FLAG_WRITE_THROUGH - do not use delayed write back

Seek System Call

```
DWORD SetFilePointer (
  HANDLE hFile,
  LONG lDistanceToMove,
  PLONG lpDistanceToMoveHigh,
  DWORD dwMoveMethod);
```

- FILE_BEGIN - distance from beginning of file
- FILE_CURRENT - distance from current position in file
- FILE_END - distance from end of file

Read And Write System Calls

```
BOOL ReadFile (
  HANDLE hFile,
  LPVOID lpBuffer,
  DWORD nNumberOfBytesToRead,
  LPDWORD lpNumberOfBytesRead,
  LPOVERLAPPED lpOverlapped);
BOOL WriteFile (
  HANDLE hFile,
  LPCVOID lpBuffer,
  DWORD nNumberOfBytesToWrite,
  LPDWORD lpNumberOfBytesWritten,
  LPOVERLAPPED lpOverlapped);

BOOL ReadFileEx (
  HANDLE hFile,
  LPVOID lpBuffer,
  DWORD nNumberOfBytesToRead,
  LPOVERLAPPED lpOverlapped,
  LPOVERLAPPED_COMPLETION_ROUTINE lpCompletionRoutine);
BOOL WriteFileEx (
  HANDLE hFile,
  LPCVOID lpBuffer,
  DWORD nNumberOfBytesToWrite,
  LPOVERLAPPED lpOverlapped,
  LPOVERLAPPED_COMPLETION_ROUTINE lpCompletionRoutine);

BOOL ReadFileScatter (
  HANDLE hFile,
  FILE_SEGMENT_ELEMENT aSegmentArray [],
  DWORD nNumberOfBytesToRead,
  LPDWORD lpReserved,
```

```
  LPOVERLAPPED lpOverlapped);
BOOL WriteFileGather (
  HANDLE hFile,
  FILE_SEGMENT_ELEMENT aSegmentArray [],
  DWORD nNumberOfBytesToWrite,
  LPDWORD lpReserved,
  LPOVERLAPPED lpOverlapped);

BOOL WINAPI GetOverlappedResult (
  HANDLE hFile,
  LPOVERLAPPED lpOverlapped,
  LPDWORD lpNumberOfBytesTransferred,
  BOOL bWait);
BOOL HasOverlappedIoCompleted (
  LPOVERLAPPED lpOverlapped);

BOOL CancelIo (HANDLE hFile);

typedef struct _OVERLAPPED {
  ULONG_PTR Internal;
  ULONG_PTR InternalHigh;
  union {
    struct {
      DWORD Offset;
      DWORD OffsetHigh;
    };
    PVOID Pointer;
  };
  HANDLE hEvent;
}
OVERLAPPED, *LPOVERLAPPED;

typedef union _FILE_SEGMENT_ELEMENT {
  PVOID64 Buffer;
  ULONGLONG Alignment;
}
FILE_SEGMENT_ELEMENT, *PFILE_SEGMENT_ELEMENT;
```

Mapped File Operations

Example: Linux Mapped File Operations

MMap And MUnmap System Calls

```
void *mmap (
  void *start, size_t length,
  int prot, int flags,
  int fd, off_t offset);
int munmap (
  void *start, size_t length);
```

- PROT_READ, PROT_WRITE, PROT_EXEC, PROT_NONE
- MAP_SHARED, MAP_PRIVATE
- MAP_FIXED
- MAP_ANONYMOUS

MRemap System Call

```
void *mremap (
  void *old_address,
  size_t old_size, size_t new_size,
  unsigned long flags);
```

- MREMAP_MAYMOVE

MSync System Call

```
int msync (
  void *start, size_t length, int flags);
```

- MS_SYNC - flush synchronously
- MS_ASYNC - flush asynchronously
- MS_INVALIDATE - invalidate other mappings

Advise System Calls

```
int posix_madvise (
  void *addr, size_t len,
  int advice);
```

- POSIX_MADV_NORMAL - no advice
- POSIX_MADV_SEQUENTIAL - sequential access
- POSIX_MADV_RANDOM - random access
- POSIX_MADV_WILLNEED - data will be used soon
- POSIX_MADV_DONTNEED - data will not be used soon

Example: Windows Mapped File Operations

CreateFileMapping System Call

```
HANDLE CreateFileMapping (
  HANDLE hFile,
  LPSECURITY_ATTRIBUTES lpFileMappingAttributes,
  DWORD flProtect,
  DWORD dwMaximumSizeHigh,
  DWORD dwMaximumSizeLow,
  LPCTSTR lpName);
```

- PAGE_READONLY, PAGE_READWRITE, PAGE_READCOPY
- SEC_COMMIT, SEC_RESERVE
- SEC_NOCACHE

- SEC_IMAGE

MapViewOfFile System Call

```
LPVOID MapViewOfFile (
  HANDLE hFileMappingObject, DWORD dwDesiredAccess,
  DWORD dwFileOffsetHigh, DWORD dwFileOffsetLow,
  DWORD dwNumberOfBytesToMap);
LPVOID MapViewOfFileEx (
  HANDLE hFileMappingObject, DWORD dwDesiredAccess,
  DWORD dwFileOffsetHigh, DWORD dwFileOffsetLow,
  DWORD dwNumberOfBytesToMap, LPVOID lpBaseAddress);
BOOL UnmapViewOfFile (
  LPCVOID lpBaseAddress);
```

- FILE_MAP_WRITE, FILE_MAP_READ, FILE_MAP_ALL_ACCESS
- FILE_MAP_COPY

Whole File Operations

Example: Linux Whole File Operations

Send File System Call

```
ssize_t sendfile (
  int out_fd,
  int in_fd,
  off_t *offset,
  size_t count);

ssize_t splice (
  int fd_in,
  loff_t *off_in,
  int fd_out,
  loff_t *off_out,
  size_t len,
  unsigned int flags);
```

Example: Windows Whole File Operations

Copy And Move System Calls

```
BOOL CopyFile (
  LPCTSTR lpExistingFileName,
  LPCTSTR lpNewFileName,
  BOOL bFailIfExists);
BOOL CopyFileEx (
  LPCTSTR lpExistingFileName,
  LPCTSTR lpNewFileName,
  LPPROGRESS_ROUTINE lpProgressRoutine,
  LPVOID lpData,
  LPBOOL pbCancel,
```

```
  DWORD dwCopyFlags);

BOOL MoveFile (
  LPCTSTR lpExistingFileName,
  LPCTSTR lpNewFileName);
BOOL MoveFileWithProgress (
  LPCTSTR lpExistingFileName,
  LPCTSTR lpNewFileName,
  LPPROGRESS_ROUTINE lpProgressRoutine,
  LPVOID lpData,
  DWORD dwFlags);

BOOL ReplaceFile (
  LPCTSTR lpReplacedFileName,
  LPCTSTR lpReplacementFileName,
  LPCTSTR lpBackupFileName,
  DWORD dwReplaceFlags,
  LPVOID lpExclude,
  LPVOID lpReserved);
```

- COPY_FILE_RESTARTABLE - store resume data in target
- REPLACEFILE_WRITE_THROUGH - do not use delayed write back

Backup System Calls

```
BOOL BackupRead (
  HANDLE hFile,
  LPBYTE lpBuffer,
  DWORD nNumberOfBytesToRead,
  LPDWORD lpNumberOfBytesRead,
  BOOL bAbort,
  BOOL bProcessSecurity,
  LPVOID* lpContext);
BOOL BackupWrite (
  HANDLE hFile,
  LPBYTE lpBuffer,
  DWORD nNumberOfBytesToWrite,
  LPDWORD lpNumberOfBytesWritten,
  BOOL bAbort,
  BOOL bProcessSecurity,
  LPVOID* lpContext);
BOOL BackupSeek (
  HANDLE hFile,
  DWORD dwLowBytesToSeek,
  DWORD dwHighBytesToSeek,
  LPDWORD lpdwLowByteSeeked,
  LPDWORD lpdwHighByteSeeked,
  LPVOID* lpContext);
```

- bAbort - last call, free the context structure
- lpContext - the context structure, allocated on first call

Directory Operations

Example: Linux Directory Operations

Directory Family Of System Calls

```
DIR *opendir (const char *name);
int closedir (DIR *dir);
struct dirent *readdir (DIR *dir);
```

Directory Family Of System Calls

```
int scandir (
  const char *dir, struct dirent ***namelist,
  int (*select) (const struct dirent *),
  int (*compar) (const struct dirent **,
                 const struct dirent **));
```

Stat System Call

```
int stat (char *path, struct stat *buf);

struct stat {
  dev_t      st_dev;      // File device
  ino_t      st_ino;      // File inode
  mode_t     st_mode;     // Access rights
  nlink_t    st_nlink;
  uid_t      st_uid;      // Owner UID
  gid_t      st_gid;      // Owner GID
  dev_t      st_rdev;     // Device ID for special files
  off_t      st_size;     // Size in bytes
  blksize_t  st_blksize;  // Block size
  blkcnt_t   st_blocks;   // Size in blocks
  time_t     st_atime;    // Last access time
  time_t     st_mtime;    // Last modification time
  time_t     st_ctime;    // Last status change time
}
```

Example: Windows Directory Operations

Directory Family Of System Calls

```
HANDLE FindFirstFile (
  LPCTSTR lpFileName,
  LPWIN32_FIND_DATA lpFindFileData);
BOOL FindNextFile (
  HANDLE hFindFile,
  LPWIN32_FIND_DATA lpFindFileData);

typedef struct _WIN32_FIND_DATA {
  DWORD dwFileAttributes;
  FILETIME ftCreationTime;
  FILETIME ftLastAccessTime;
  FILETIME ftLastWriteTime;
  DWORD    nFileSizeHigh;
  DWORD    nFileSizeLow;
```

```
  DWORD    dwReserved0;
  DWORD    dwReserved1;
  TCHAR    cFileName [MAX_PATH (= 260)];
  TCHAR    cAlternateFileName [14];
} WIN32_FIND_DATA;
```

Unique File System Calls

```
UINT GetTempFileName (
  LPCTSTR lpPathName,
  LPCTSTR lpPrefixString,
  UINT uUnique,
  LPTSTR lpTempFileName
);
```

Sharing Support

Example: Linux Sharing Operations

File Locking System Call

```
int flock (int fd, int operation);
```

- LOCK_EX - exclusive advisory lock
- LOCK_SH - shared advisory lock
- LOCK_UN - advisory unlock

Block Locking System Call

```
int fcntl (int fd, int cmd, struct flock *lock);
struct flock {
  ...
  short l_type;    // F_WRLCK, F_RDLCK, F_UNLCK
  short l_whence;  // SEEK_SET, SEEK_CUR, SEEK_END
  off_t l_start;   // Starting offset for lock
  off_t l_len;     // Number of bytes to lock
  pid_t l_pid;     // Who blocks our lock
  ...
};
```

- cmd - F_GETLK, F_SETLK, F_SETLKW - advisory or mandatory lock get, set, set with wait
- l_type - exclusive, shared, unlock
- l_whence - how to interpret starting offset

Example: Windows Sharing Operations

Block Locking System Calls

```
BOOL LockFile (
  HANDLE hFile,
  DWORD dwFileOffsetLow,
  DWORD dwFileOffsetHigh,
  DWORD nNumberOfBytesToLockLow,
  DWORD nNumberOfBytesToLockHigh);
BOOL UnlockFile (
  HANDLE hFile,
  DWORD dwFileOffsetLow,
  DWORD dwFileOffsetHigh,
  DWORD nNumberOfBytesToUnlockLow,
  DWORD nNumberOfBytesToUnlockHigh);
```

Consistency Support

Example: Windows Transaction Operations

Transaction System Calls

```
HANDLE CreateTransaction (
  LPSECURITY_ATTRIBUTES lpTransactionAttributes,
  LPGUID UOW,
  DWORD CreateOptions,
  DWORD IsolationLevel,
  DWORD IsolationFlags,
  DWORD Timeout,
  LPWSTR Description);

BOOL CommitTransaction (
  HANDLE TransactionHandle);
BOOL RollbackTransaction (
  HANDLE TransactionHandle);

HANDLE CreateFileTransacted (
  LPCTSTR lpFileName,
  DWORD dwDesiredAccess,
  DWORD dwShareMode,
  LPSECURITY_ATTRIBUTES lpSecurityAttributes,
  DWORD dwCreationDisposition,
  DWORD dwFlagsAndAttributes,
  HANDLE hTemplateFile,
  HANDLE hTransaction,
  PUSHORT pusMiniVersion,
  PVOID pExtendedParameter);

BOOL DeleteFileTransacted(
  LPCTSTR lpFileName,
  HANDLE hTransaction);

BOOL CreateDirectoryTransacted (...);
BOOL RemoveDirectoryTransacted (...);

BOOL MoveFileTransacted (...);
BOOL CopyFileTransacted (...);
```

File Subsystem Internals

Disk Layout

Example: FAT File System

FAT Boot Sector

```
Offset  Length  Contents
--------------------------------------------------------------------------------
00h     3       Boot loader code
03h 8  System vendor ID
0Bh 2  Bytes per sector
0Dh 1  Sectors per cluster
0Eh 2  Number of reserved sectors before FAT
10h 1  Number of FAT copies
11h 2  Number of root entries
13h 2  Number of sectors for small partitions
15h 1  Media descriptor
16h 2  Sectors per FAT
18h 2  Sectors per head
1Ah 2  Heads per cylinder
1Ch 4  Number of hidden sectors before boot
20h 4  Number of sectors for large partitions
24h     474     Boot loader code
1FEh    2       Magic (0AA55h)

24h     4       Number of sectors per FAT
28h 2  File system flags
2Ah 2  File system version
2Ch 4  First cluster of root directory
30h 2  Sector with file system information
32h 2  Sector with boot sector copy
34h 12   Reserved
40h     1       Physical drive number
41h     1       Reserved
42h     1       Magic (28h or 29h)
43h     4       Volume serial number
47h     11      Volume label
52h     8       File system ID
5Ah     420     Boot loader code
```

FAT Directory Entry

```
Offset  Length  Contents
--------------------------------------------------------------------------------
00h 8 File name padded with spaces or 00h or 1Eh
08h 3 File extension padded with spaces
0Bh 1 File attributes (archive, dir, vol, sys, hidden, read only)
0Ch 10 Reserved
16h 2 Last modification time
18h 2 Last modification date
1Ah 2 Starting cluster
1Ch 4 File size

0Ch 1 Reserved
0Dh 3       Creation time
10h 2 Creation date
12h 2 Last access date
14h 2 Starting cluster high word
```

```
00h 1 Sequential number of long name fragment and last fragment flag
01h 10 Long name (5 characters UNICODE)
0Bh 1 File attributes (0Fh means vol, sys, hidden, read only)
0Ch 1 Reserved
0Dh 1 Checksum of short name
0Eh 12 Long name (6 characters UNICODE)
1Ah 2 Reserved
1Ch 4 Long name (2 characters UNICODE)
```

Example: HPFS File System

HPFS F-Node

Offset	Length	Contents
00h	4	Magic (0F7E40AAEh)
04h	4	Sequential read history (not implemented)
08h	4	Fast read history (not implemented)
0Ch	1	Name length
0Dh	15	Last 15 characters of name
1Ch	4	Sector of containing directory
20h	4	Size of external access control list
24h	4	Sector of external access control list
28h	2	Size of internal access control list
2Ah	1	Indicates whether access control list is large tree
2Bh	1	History bit count (not implemented)
2Ch	4	Size of external extended attributes
30h	4	Sector of external extended attributes
34h	2	Size of internal extended attributes
36h	1	Indicates whether extended attributes is large tree
37h	1	Directory or file F node flag
38h	1	Indicates whether runs are large tree
39h	3	Padding
3Ch	1	Number of free array entries
3Dh	1	Number of used array entries
3Eh	2	Offset of first free array entry
40h	4	Offset of this run in file
44h	4	Number of sectors in run
48h	4	Starting sector of this run
40h	4	Offset of this subtree in file
44h	4	Starting sector of this subtree
0A0h	4	File length
0A4h	4	Number of vital extended attributes
0A8h	16	User identity (not implemented)
0B8h	2	Offset of first access control list or extended attribute entry
0BAh	10	Reserved
0C4h	316	Access control list and extended attribute entries

HPFS Directory Entry

Offset	Length	Contents
00h	4	Magic (77E40AAEh)
04h	4	Offset of first free directory entry
08h	4	Tree root indication
0Ch	4	Sector of tree parent block
10h	4	Sector of this tree node block

```
14h     2       Directory entry size
16h     1       Flags
17h     1       Attributes (read only, hidden, sys, dir, archive, long name)
18h     4       Sector of F node
1Ch     4       Last modification time
20h     4       File size
24h     4       Last access time
28h     4       Creation time
2Ch     4       Size of extended attributes in F node
30h     1       Size of access control list
31h     1       Code page index
32h     1       Name size
33h     X       Name
33h+X   X       Access control list
                Padding to multiple of 4 bytes
```

Example: EXT2 And EXT3 And EXT4 File Systems

EXT2 Inode Structure

```
struct ext2_inode {
  __u16  i_mode;   /* File mode */
  __u16  i_uid;                         /* Owner ID */
  __u32  i_size;                        /* Size in bytes */
  __u32  i_atime;                       /* Access time */
  __u32  i_ctime;                       /* Creation time */
  __u32  i_mtime;                       /* Modification time */
  __u32  i_dtime;                       /* Deletion Time */
  __u16  i_gid;                         /* Group ID */
  __u16  i_links_count;                 /* Links count */
  __u32  i_blocks;                      /* Blocks count */
  __u32  i_flags;                       /* File flags */
  __u32  i_block [EXT2_N_BLOCKS];       /* Ptrs to blocks */
  __u32  i_version;                     /* File version for NFS */
  __u32  i_file_acl;                    /* File ACL */
  __u32  i_dir_acl;                     /* Directory ACL */
  __u32  i_faddr;                       /* Fragment address */
  __u8   l_i_frag;                      /* Fragment number */
  __u8   l_i_fsize;                     /* Fragment size */
};

#define EXT2_DIR_BLOCKS  12
#define EXT2_IND_BLOCK          EXT2_DIR_BLOCKS
#define EXT2_DIND_BLOCK  (EXT2_IND_BLOCK + 1)
#define EXT2_TIND_BLOCK  (EXT2_DIND_BLOCK + 1)
#define EXT2_N_BLOCKS           (EXT2_TIND_BLOCK + 1)

#define EXT2_SECRM_FL    0x00000001      /* Secure del */
#define EXT2_SYNC_FL     0x00000008      /* Sync update */
#define EXT2_IMMUTABLE_FL       0x00000010      /* Immutable */
#define EXT2_APPEND_FL          0x00000020      /* Only ap */
```

EXT2 Directory Entry Structure

```
struct ext2_dir_entry_2 {
  __u32  inode;               /* Inode number */
  __u16  rec_len;                      /* Directory entry length */
  __u8   name_len;                     /* Name length */
  __u8   file_type;   /* File type */
  char   name [EXT2_NAME_LEN];    /* File name */
};

#define EXT2_NAME_LEN 255

#define EXT2_FT_REG_FILE        1
#define EXT2_FT_DIR             2
#define EXT2_FT_CHRDEV   3
#define EXT2_FT_BLKDEV   4
#define EXT2_FT_SYMLINK  7
```

EXT2 I-Node Usage

```
> df -i /
Filesystem            Inodes   IUsed   IFree    IUse% Mounted on
/dev/hda1            24903680 1007827 23895853    5%      /
```

EXT2 Superblock Information

```
> tune2fs -l /dev/hda1
tune2fs 1.35 (28-Feb-2004)
Filesystem volume name:   /
Last mounted on:          (not available)
Filesystem UUID:          7404a4b8-84f5-11d6-9629-99bdda41ad84
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal ext_attr dir_index filetype
                          needs_recovery sparse_super large_file
Default mount options:    (none)
Filesystem state:         clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              24903680
Block count:              49785427
Reserved block count:     1024
Free blocks:              8828311
Free inodes:              23896298
First block:              0
Block size:               4096
Fragment size:            4096
Blocks per group:         32768
Fragments per group:      32768
Inodes per group:         16384
Inode blocks per group:   512
Last mount time:          Tue Mar 15 21:21:31 2005
Last write time:          Tue Mar 15 21:21:31 2005
Mount count:              28
Maximum mount count:      -1
Last checked:             Wed Dec 22 11:55:23 2004
Check interval:           0 (none)
Reserved blocks uid:      0 (user root)
Reserved blocks gid:      0 (group root)
First inode:              11
Inode size:               128
Journal inode:            8
```

```
First orphan inode:       328593
Default directory hash:   tea
Directory Hash Seed:      b099544d-7257-456c-8666-4c646f123e16
Journal backup:           inode blocks
```

Example: NTFS File System

MFT Entry

```
typedef struct MftEntry
{
   char         Signature [4];          // Magic "FILE"
   ushort       FixupOffset;
   ushort       FixupSize;
   ulong
   ulong
   ushort       Sequence;               // Sequence number in MFT
   ushort       HardLinks;              // Hard link count
   ushort       AttribOffset;           // Offset of attributes
   ushort       Flags;
   ulong        RecLength;              // True record size
   ulong        AllLength;              // Allocated record size
   cluster      BaseMftRec;             // Base entry of this entry or 0
   ushort       MinIdentificator;
   ushort       FixupPattern;
   ushort       FixupList [];
};


typedef struct AttributeEntry
{
   ulong        Type;                   // Type of attribute
   ushort       Length;                 // Length of attribute entry
   ushort
   byte         Residency;              // Resident or nonresident
   byte         NameLen;                // Length of name
   ushort       Offset;                 // Offset of name or data
   byte         Compressed;             // Compressed or uncompressed
   byte
   ushort       Identificator;
   union
   {
      ResidentAttribEntry Resident;
      NonresidentAttribEntry NonResident;
   };
};

typedef struct ResidentAttribEntry
{
   ushort       Size;                   // Size of attribute
   ushort
   ushort       Offset;                 // Offset of value
   ushort       IndexFlag;
};

typedef struct NonResidentAttribEntry
{
   cluster      SegFirst;               // First cluster in this segment
   cluster      SegLast;                // Last cluster in this segment
   USHORT       Offset;                 // Offset of run list
   USHORT       ComprEngine;            // Compression engine
   USHORT
```

```
   USHORT
   XLONG        Allocated;              // Allocated disk space
   XLONG        Size;                   // Size of uncompressed attribute
   XLONG        Compressed;             // Size of compressed attribute
};


typedef struct DirectoryEntry
{
   xlong        RecordNumber;           // This MFT record number
   ushort       Length;                 // Length of directory entry
   ushort
   byte         Flags;
   byte
   xlong        Parent;                 // Parent MFT record number
   time         Create;                 // Creation time
   time         ModifyFile;             // Last file modification time
   time         ModifyEntry;            // Last entry modification time
   time         Access;                 // Last access time
   xlong        Alloc;                  // Allocated size
   xlong        Size;                   // Real size
   xlong
   byte         NameLen;                // Name length in words
   byte         NameType;               // Type of filename
   ushort       Filename [];            // Variable length filename
};
```

Multiple Streams

Stream Enumeration System Calls

```
HANDLE FindFirstStreamW (
  LPCWSTR lpFileName,
  STREAM_INFO_LEVELS InfoLevel,
  LPVOID lpFindStreamData,
  DWORD dwFlags);
BOOL FindNextStreamW (
  HANDLE hFindStream,
  LPVOID lpFindStreamData);

typedef enum _STREAM_INFO_LEVELS {
  FindStreamInfoStandard
} STREAM_INFO_LEVELS;

typedef struct _WIN32_FIND_STREAM_DATA {
  LARGE_INTEGER StreamSize;
  WCHAR cStreamName [MAX_PATH + 36];
} WIN32_FIND_STREAM_DATA;
```

Example: CD File System

ISO9660 Primary Volume Descriptor

```
Offset  Length  Contents
-------------------------------------------------------------------------
0       8       Magic (1, "CD001", 1, 0)
8       32      System identifier
28h     32      Volume identifier
```

```
48h     8       Zero
50h     8       Number of sectors (both endian dword)
58h     32      Zero
78h     4       Volume set size (1, both endian word)
7Ch     4       Volume sequence number (1, both endian word)
80h     4       Sector size (2048, both endian word)
        8       Path table length in bytes
        4       Number of first sector in first little endian path table
        4       Number of first sector in second little endian path table or 0
        4       Number of first sector in first big endian path table
        4       Number of first sector in second big endian path table or 0
        34      Root directory record
        128     Volume set identifier
        128     Publisher identifier
        128     Data preparer identifier
        128     Application identifier
        37      Copyright file identifier
        37      Abstract file identifier
        37      Bibliographical file identifier
        17      Date and time of volume creation
        17      Date and time of most recent modification
        17      Date and time when volume expires
        17      Date and time when volume is effective
        1       1
        1       0
        512     Reserved for application use
        653     Zero
```

ISO9660 Directory Entry

```
Offset  Length  Contents
------------------------------------------------------------------------------
        1       Record size in bytes (must be even)
        1       Number of sectors in extended attribute record
        8       Number of the first sector of file data or directory
        8       Number of bytes of file data or length of directory
        7       Date and time
        1       Flags (HID, DIR ..., LAST for multiple sections)
        1       Interleaved file unit size or 0
        1       Interleaved file gap size or 0
        4       Volume sequence number (1)
        1       Length of name
        N       Name
        P       Padding
```

Integration Of Multiple File Subsystems

Example: Linux Virtual File System

Linux VFS Layer

- Purpose

  Unified interface to user processes

  Kernel abstraction for different implementations
- Function

Service file/file system syscalls
Manage file/file system data structures
Provide generic routines for common operations
Interact with file system implementations

Basic VFS Objects

- superblock
  Mounted instance of a file system.
- dentry
  Directory entry in the filesystem hierarchy.
- inode
  In-kernel representation of a file.
- file
  Open file descriptor.

Super Block

- Handles metadata
  Retrieves and stores metadata from media
- Holds file system instance data
  device, block size, dirty flags
  root inode, list of dirty inodes
  superblock operations

Super Block Structure

```
struct super_block {
  struct list_head        s_list;         /* Keep this first */
  dev_t                   s_dev;          /* search index; _not_ kdev_t */

  unsigned long           s_blocksize;
  unsigned long           s_old_blocksize;
  unsigned char           s_blocksize_bits;
  unsigned char           s_dirt;
  unsigned long long      s_maxbytes;     /* Max file size */

  struct file_system_type *s_type;
  struct super_operations *s_op;
  struct dquot_operations *dq_op;
  struct quotactl_ops     *s_qcop;
  struct export_operations *s_export_op;

  unsigned long           s_flags;
  unsigned long           s_magic;
  struct dentry           *s_root;
  struct rw_semaphore     s_umount;
  struct semaphore        s_lock;
  int                     s_count;
  int                     s_syncing;
  int                     s_need_sync_fs;
```

```
  atomic_t                s_active;
  void                    *s_security;
  struct xattr_handler    **s_xattr;

  struct list_head        s_inodes;       /* all inodes */
  struct list_head        s_dirty;        /* dirty inodes */
  struct list_head        s_io;           /* parked for writeback */
  struct hlist_head       s_anon;         /* anonymous dentries for (nfs) exporting */
  struct list_head        s_files;

  struct block_device     *s_bdev;
  struct list_head        s_instances;
  struct quota_info       s_dquot;        /* Diskquota specific options */

  int                     s_frozen;
  wait_queue_head_t       s_wait_unfrozen;

  char s_id[32];                          /* Informational name */

  void                    *s_fs_info;     /* Filesystem private info */

  /*
   * The next field is for VFS *only*. No filesystems have any business
   * even looking at it. You had been warned.
   */
  struct semaphore s_vfs_rename_sem;      /* Kludge */

  /* Granuality of c/m/atime in ns.
     Cannot be worse than a second */
  u32                s_time_gran;
};
```

Super Block Operations

```
struct super_operations {
  struct inode *(*alloc_inode) (struct super_block *sb);
  void (*destroy_inode) (struct inode *);

  void (*read_inode) (struct inode *);

  void (*dirty_inode) (struct inode *);
  int (*write_inode) (struct inode *, int);
  void (*put_inode) (struct inode *);
  void (*drop_inode) (struct inode *);
  void (*delete_inode) (struct inode *);

  void (*put_super) (struct super_block *);
  void (*write_super) (struct super_block *);

  int (*sync_fs)(struct super_block *sb, int wait);
  void (*write_super_lockfs) (struct super_block *);
  void (*unlockfs) (struct super_block *);

  int (*statfs) (struct super_block *, struct kstatfs *);
  int (*remount_fs) (struct super_block *, int *, char *);

  void (*clear_inode) (struct inode *);

  void (*umount_begin) (struct super_block *);

  int (*show_options) (struct seq_file *, struct vfsmount *);

  ssize_t (*quota_read) (
    struct super_block *, int, char *, size_t, loff_t);
```

```
  ssize_t (*quota_write) (
    struct super_block *, int, const char *, size_t, loff_t);
};
```

Directory Entry

- Name to inode translation structure
- Cached agressively by the VFS (dcache)
- Eliminates private FS lookups, caching
- Negative dentries

Directory Entry Structure

```
struct dentry {
  atomic_t d_count;
  unsigned int d_flags;           /* protected by d_lock */
  spinlock_t d_lock;              /* per dentry lock */

  struct inode *d_inode;          /* Where the name belongs to - NULL is
                                   * negative */
  /*
   * The next three fields are touched by __d_lookup.  Place them here
   * so they all fit in a 16-byte range, with 16-byte alignment.
   */
  struct dentry *d_parent;        /* parent directory */
  struct qstr d_name;

  struct list_head d_lru;         /* LRU list */
  struct list_head d_child;       /* child of parent list */
  struct list_head d_subdirs;     /* our children */
  struct list_head d_alias;       /* inode alias list */

  unsigned long d_time;           /* used by d_revalidate */
  struct dentry_operations *d_op;
  struct super_block *d_sb;       /* The root of the dentry tree */
  void *d_fsdata;                 /* fs-specific data */

  struct rcu_head d_rcu;
  struct dcookie_struct *d_cookie; /* cookie, if any */
  struct hlist_node d_hash;       /* lookup hash list */

  int d_mounted;

  unsigned char d_iname[DNAME_INLINE_LEN_MIN];    /* small names */
};
```

Directory Entry Operations

```
struct dentry_operations {
  int (*d_revalidate) (struct dentry *, struct nameidata *);
  int (*d_hash) (struct dentry *, struct qstr *);
  int (*d_compare) (struct dentry *, struct qstr *, struct qstr *);
  int (*d_delete) (struct dentry *);
  void (*d_release) (struct dentry *);
  void (*d_iput) (struct dentry *, struct inode *);
};
```

Index Node

- VFS abstraction for a file
- Held in inode cache by VFS
- Hold inode and default file operations
- Contain FS specific areas

Index Node Structure

```
struct inode {
  struct hlist_node       i_hash;
  struct list_head        i_list;
  struct list_head        i_sb_list;
  struct list_head        i_dentry;

  unsigned long           i_ino;
  atomic_t                i_count;
  umode_t                 i_mode;
  unsigned int            i_nlink;

  uid_t                   i_uid;
  gid_t                   i_gid;
  dev_t                   i_rdev;
  loff_t                  i_size;

  struct timespec         i_atime;
  struct timespec         i_mtime;
  struct timespec         i_ctime;

  unsigned int            i_blkbits;
  unsigned long           i_blksize;

  unsigned long           i_version;
  unsigned long           i_blocks;
  unsigned short          i_bytes;

  unsigned char           i_sock;
  spinlock_t              i_lock; /* i_blocks, i_bytes, maybe i_size */
  struct semaphore        i_sem;
  struct rw_semaphore     i_alloc_sem;

  struct inode_operations *i_op;
  struct file_operations  *i_fop; /* former ->i_op->default_file_ops */

  struct super_block      *i_sb;
  struct file_lock        *i_flock;

  struct address_space    *i_mapping;
  struct address_space    i_data;
#ifdef CONFIG_QUOTA
  struct dquot            *i_dquot[MAXQUOTAS];
#endif

  /* These three should probably be a union */
  struct list_head        i_devices;
  struct pipe_inode_info  *i_pipe;
  struct block_device     *i_bdev;
  struct cdev             *i_cdev;
  int                     i_cindex;

  __u32                   i_generation;
```

```
#ifdef CONFIG_DNOTIFY
  unsigned long           i_dnotify_mask; /* Directory notify events */
  struct dnotify_struct   *i_dnotify; /* for directory notifications */
#endif

  unsigned long           i_state;
  unsigned long           dirtied_when;   /* jiffies of first dirtying */

  unsigned int            i_flags;

  atomic_t                i_writecount;
  void                    *i_security;
  union {
    void                  *generic_ip;
  } u;
#ifdef __NEED_I_SIZE_ORDERED
  seqcount_t              i_size_seqcount;
#endif
};
```

Index Node Operations

```
struct inode_operations {
  int (*create) (struct inode *, struct dentry *, int, struct nameidata *);
  struct dentry * (*lookup) (
    struct inode *, struct dentry *, struct nameidata *);

  int (*link) (struct dentry *,struct inode *,struct dentry *);
  int (*unlink) (struct inode *,struct dentry *);

  int (*symlink) (struct inode *,struct dentry *,const char *);

  int (*mkdir) (struct inode *,struct dentry *,int);
  int (*rmdir) (struct inode *,struct dentry *);

  int (*mknod) (struct inode *,struct dentry *,int,dev_t);
  int (*rename) (
    struct inode *, struct dentry *, struct inode *, struct dentry *);

  int (*readlink) (struct dentry *, char __user *,int);
  int (*follow_link) (struct dentry *, struct nameidata *);
  void (*put_link) (struct dentry *, struct nameidata *);

  void (*truncate) (struct inode *);

  int (*permission) (struct inode *, int, struct nameidata *);

  int (*setattr) (struct dentry *, struct iattr *);
  int (*getattr) (struct vfsmount *mnt, struct dentry *, struct kstat *);
  int (*setxattr) (struct dentry *, const char *,const void *,size_t,int);
  ssize_t (*getxattr) (struct dentry *, const char *, void *, size_t);
  ssize_t (*listxattr) (struct dentry *, char *, size_t);
  int (*removexattr) (struct dentry *, const char *);
};
```

File Structure

```
struct file {
  struct list_head        f_list;
  struct dentry           *f_dentry;
  struct vfsmount         *f_vfsmnt;
  struct file_operations  *f_op;
  atomic_t                f_count;
  unsigned int            f_flags;
  mode_t                  f_mode;
  int                     f_error;
  loff_t                  f_pos;
  struct fown_struct      f_owner;
  unsigned int            f_uid, f_gid;
  struct file_ra_state    f_ra;

  size_t                  f_maxcount;
  unsigned long           f_version;
  void                    *f_security;

  /* needed for tty driver, and maybe others */
  void                    *private_data;

#ifdef CONFIG_EPOLL
  /* Used by fs/eventpoll.c to link all the hooks to this file */
  struct list_head        f_ep_links;
  spinlock_t              f_ep_lock;
#endif /* #ifdef CONFIG_EPOLL */
  struct address_space    *f_mapping;
};
```

File Operations

```
struct file_operations {
  struct module *owner;

  loff_t (*llseek) (struct file *, loff_t, int);
  ssize_t (*read) (struct file *, char __user *, size_t, loff_t *);
  ssize_t (*aio_read) (struct kiocb *, char __user *, size_t, loff_t);
  ssize_t (*write) (struct file *, const char __user *, size_t, loff_t *);
  ssize_t (*aio_write) (struct kiocb *, const char __user *, size_t, loff_t);

  int (*readdir) (struct file *, void *, filldir_t);
  unsigned int (*poll) (struct file *, struct poll_table_struct *);

  int (*ioctl) (struct inode *, struct file *, unsigned int, unsigned long);
  long (*unlocked_ioctl) (struct file *, unsigned int, unsigned long);
  long (*compat_ioctl) (struct file *, unsigned int, unsigned long);

  int (*mmap) (struct file *, struct vm_area_struct *);
  int (*open) (struct inode *, struct file *);
  int (*flush) (struct file *);
  int (*release) (struct inode *, struct file *);

  int (*fsync) (struct file *, struct dentry *, int datasync);
  int (*aio_fsync) (struct kiocb *, int datasync);
  int (*fasync) (int, struct file *, int);
  int (*lock) (struct file *, int, struct file_lock *);

  ssize_t (*readv) (
    struct file *, const struct iovec *, unsigned long, loff_t *);
  ssize_t (*writev) (
    struct file *, const struct iovec *, unsigned long, loff_t *);
```

```
  ssize_t (*sendfile) (
    struct file *, loff_t *, size_t, read_actor_t, void *);
  ssize_t (*sendpage) (
    struct file *, struct page *, int, size_t, loff_t *, int);

  unsigned long (*get_unmapped_area) (
    struct file *, unsigned long, unsigned long,
    unsigned long, unsigned long);

  int (*check_flags)(int);
  int (*dir_notify)(struct file *filp, unsigned long arg);
  int (*flock) (struct file *, int, struct file_lock *);
};
```

Auxiliary VFS Objects

- filesystem type

  Filesystem constructor/destructor.
- vfsmount

  A subtree in the filesystem hierarchy.
- nameidata

  A result of a directory lookup.
- address space

  Mapping between file and disk blocks.

File System Type Structure

```
struct file_system_type {
  const char *name;
  int fs_flags;

  struct super_block *(*get_sb) (
    struct file_system_type *, int, const char *, void *);
  void (*kill_sb) (struct super_block *);

  struct module *owner;
  struct file_system_type * next;
  struct list_head fs_supers;
};
```

File System Type Generics

```
struct super_block * get_sb_xxx (
  struct file_system_type *fs_type,
  int flags, const char *dev_name, void *data,
  int (*fill_super) (struct super_block *, void *, int))
```

- get_sb_nodev ()
- get_sb_bdev ()
- get_sb_single ()
- get_sb_pseudo ()

```
void kill_xxx_super (struct super_block * sb)
```

- kill_block_super ()
- kill_anon_super ()
- kill_litter_super ()

VFS Mount Structure

```
struct vfsmount {
  struct list_head mnt_hash;

  struct vfsmount *mnt_parent;    /* fs we are mounted on */

  struct dentry *mnt_mountpoint;  /* dentry of mountpoint */
  struct dentry *mnt_root;        /* root of the mounted tree */

  struct super_block *mnt_sb;     /* pointer to superblock */

  struct list_head mnt_mounts;    /* list of children, anchored here */
  struct list_head mnt_child;     /* and going through their mnt_child */

  atomic_t mnt_count;
  int mnt_flags;
  int mnt_expiry_mark;            /* true if marked for expiry */
  char *mnt_devname;              /* Name of device e.g. /dev/dsk/hda1 */

  struct list_head mnt_list;
  struct list_head mnt_fslink;    /* link in fs-specific expiry list */

  struct namespace *mnt_namespace; /* containing namespace */
};
```

Directory Lookup Result Structure

```
struct nameidata {
  struct dentry   *dentry;
  struct vfsmount *mnt;
  struct qstr     last;
  unsigned int    flags;
  int             last_type;
  unsigned        depth;
  char *saved_names[MAX_NESTED_LINKS + 1];

  /* Intent data */
  union {
          struct open_intent open;
  } intent;
};
```

Address Space Structure

```
struct address_space {
  struct inode            *host;          /* owner: inode, block_device */

  struct radix_tree_root  page_tree;      /* radix tree of all pages */
  spinlock_t              tree_lock;      /* and spinlock protecting it */
```

```
  unsigned int            i_mmap_writable;/* count VM_SHARED mappings */
  struct prio_tree_root   i_mmap;         /* tree of private and shared mappings */
  struct list_head        i_mmap_nonlinear;/*list VM_NONLINEAR mappings */
  spinlock_t              i_mmap_lock;    /* protect tree, count, list */

  unsigned int            truncate_count; /* Cover race condition with truncate */
  unsigned long           nrpages;        /* number of total pages */
  pgoff_t                 writeback_index;/* writeback starts here */

  struct address_space_operations *a_ops; /* methods */

  unsigned long           flags;          /* error bits/gfp mask */
  struct backing_dev_info *backing_dev_info; /* device readahead, etc */

  spinlock_t              private_lock;   /* for use by the address_space */
  struct list_head        private_list;   /* ditto */
  struct address_space    *assoc_mapping; /* ditto */
};
```

Address Space Operations

```
struct address_space_operations {
  int (*writepage) (struct page *page, struct writeback_control *wbc);
  int (*readpage) (struct file *, struct page *);
  int (*sync_page) (struct page *);

  /* Write back some dirty pages from this mapping. */
  int (*writepages) (struct address_space *, struct writeback_control *);

  /* Set a page dirty */
  int (*set_page_dirty) (struct page *page);

  int (*readpages) (
    struct file *filp, struct address_space *mapping,
    struct list_head *pages, unsigned nr_pages);

  /*
   * ext3 requires that a successful prepare_write() call be followed
   * by a commit_write() call - they must be balanced
   */
  int (*prepare_write) (struct file *, struct page *, unsigned, unsigned);
  int (*commit_write) (struct file *, struct page *, unsigned, unsigned);

  /* Unfortunately this kludge is needed for FIBMAP. Don't use it */
  sector_t (*bmap) (struct address_space *, sector_t);

  int (*invalidatepage) (struct page *, unsigned long);
  int (*releasepage) (struct page *, int);
  ssize_t (*direct_IO) (
    int, struct kiocb *, const struct iovec *iov,
    loff_t offset, unsigned long nr_segs);
};
```

References

References

1. Braam, P. J.: Linux Virtual File System, http://www.coda.cs.cmu.edu/doc/talks/linuxvfs, Apr 1998

2. Brown, N.: The Linux Virtual File-system Layer, http://www.cse.unsw.edu.au/~neilb/oss/linux-commentary/vfs.html, Dec 1999

3. Aivazian, T.: Linux Kernel 2.4 Internals, http://www.faqs.org/docs/kernel_2_4/lki.html, Aug 2002

4. Brouwer, A.: The Linux Kernel, http://www.win.tue.nl/~aeb/linux/lk/lk.html, Jan 2003

5. Zhang, Y.: Linux Kernel Programming, http://www.cs.utexas.edu/users/ygz/378-03S, Apr 2003

6. Kiran, R.: Writing a Simple Filesystem, http://www.geocities.com/ravikiran_uvs/articles/rkfs.html

7. Corbet, J.: Creating Linux Virtual Filesystems, http://lwn.net/Articles/57369, Nov 2003

Chapter 6. Network Subsystem

Abstractions And Operations

Sockets

Socket System Call

```
int socket (int domain, int type, int protocol);
```

Domain specifies socket protocol class:

- PF_UNIX - local communication
- PF_INET - IPv4 protocol family
- PF_INET6 - IPv6 protocol family
- PF_IPX - IPX protocol family
- PF_NETLINK - kernel communication
- PF_PACKET - raw packet communication

Type specifies socket semantics:

- SOCK_STREAM - reliable bidirectional ordered stream
- SOCK_RDM - reliable bidirectional unordered messages
- SOCK_DGRAM - unreliable bidirectional unordered messages
- SOCK_SEQPACKET - reliable bidirectional ordered messages
- SOCK_RAW - raw packets

Protocol specifies socket protocol:

- 0 - class and type determine protocol
- other - identification of supported protocol

Bind System Call

```
int bind (int sockfd, struct sockaddr *my_addr, socklen_t addrlen);

#define __SOCKADDR_COMMON(sa_prefix) \
  sa_family_t sa_prefix##family

struct sockaddr_in
{
  __SOCKADDR_COMMON (sin_);
  in_port_t      sin_port;
  struct in_addr sin_addr;
  unsigned char  sin_zero [sizeof (struct sockaddr) -
                           __SOCKADDR_COMMON_SIZE -
```

```
                              sizeof (in_port_t) -
                              sizeof (struct in_addr)];
};

struct sockaddr_in6
{
  __SOCKADDR_COMMON (sin6_);
  in_port_t       sin6_port;
  uint32_t        sin6_flowinfo;
  struct in6_addr sin6_addr;
  uint32_t        sin6_scope_id;
};
```

Listen System Call

```
int listen (int sockfd, int backlog);
```

Accept System Call

```
int accept (int sockfd, struct sockaddr *addr, socklen_t *addrlen);
```

Connect System Call

```
int connect (int sockfd,
             const struct sockaddr *serv_addr,
             socklen_t addrlen);
```

Send Family Of System Calls

```
ssize_t send (int sockfd, const void *buf, size_t len, int flags);
ssize_t sendto (int sockfd, const void *buf, size_t len, int flags,
                const struct sockaddr *to, socklen_t tolen);
ssize_t sendmsg (int sockfd, const struct msghdr *msg, int flags);

struct msghdr
{
  void         *msg_name;       // optional address
  socklen_t    msg_namelen;     // optional address length
  struct iovec *msg_iov;        // array for scatter gather
  size_t       msg_iovlen;      // array for scatter gather length
  void         *msg_control;    // additional control data
  socklen_t    msg_controllen;  // additional control data length
  int          msg_flags;
};
```

Recv Family Of System Calls

```
ssize_t recv (int sockfd, void *buf, size_t len, int flags);
ssize_t recvfrom (int sockfd, void *buf, size_t len, int flags,
                  struct sockaddr *from, socklen_t *fromlen);
ssize_t recvmsg (int sockfd, struct msghdr *msg, int flags);

struct msghdr
{
  void         *msg_name;       // optional address
```

```
  socklen_t    msg_namelen;     // optional address length
  struct iovec *msg_iov;        // array for scatter gather
  size_t       msg_iovlen;      // array for scatter gather length
  void         *msg_control;    // additional control data
  socklen_t    msg_controllen;  // additional control data length
  int          msg_flags;
};
```

Select And Poll System Calls

```
int select (int setsize,
            fd_set *readfds,
            fd_set *writefds,
            fd_set *exceptfds,
            struct timeval *timeout);

int poll (struct pollfd *ufds,
          unsigned int nfds,
          int timeout);

struct pollfd
{
  int fd;
  short events;         // requested events
  short revents;        // returned events
};
```

Socket Options System Calls

```
int getsockopt (int sockfd, int level,
                int optname, void *optval, socklen_t *optlen);

int setsockopt (int sockfd, int level,
                int optname, const void *optval, socklen_t optlen);
```

Example: Unix Sockets

Socket Address Structure

```
struct sockaddr_un
{
  sa_family_t  sun_family;              // set to AF_UNIX
  char         sun_path [PATH_MAX];     // socket name
};
```

Socket Pair System Call

```
int socketpair (int domain,
                int type,
                int protocol,
                int sockets [2]);
```

Example: Sending File Descriptor Array

```
struct msghdr message;
int decriptors [LEN];
char buffer [CMSG_SPACE (sizeof (descriptors))];

// Prepare message header.
message.msg_control = buffer;
message.msg_controllen = sizeof (buffer);
struct cmsghdr *control = CMSG_FIRSTHDR (&message);

// Fill in protocol identifier and protocol specific type.
// Use of SOL_SOCKET instead of AF_UNIX for legacy reasons.
control->cmsg_level = SOL_SOCKET;
control->cmsg_type = SCM_RIGHTS;
control->cmsg_len = CMSG_LEN (sizeof (int) * LEN);

// Fill in protocol specific payload.
memcpy (CMSG_DATA (control), descriptors, sizeof (descriptors));

// Complete message header.
message->msg_controllen = control->cmsg_len;
```

Socket Listing

```
> netstat --unix --all (servers and established)
Proto RefCnt Flags   Type   State     Path
unix  2      [ ACC ] STREAM LISTENING /var/run/acpid.socket
unix  2      [ ACC ] STREAM LISTENING /tmp/.font-unix/fs7100
unix  2      [ ACC ] STREAM LISTENING /tmp/.gdm_socket
unix  2      [ ACC ] STREAM LISTENING /tmp/.X11-unix/X0
unix  2      [ ACC ] STREAM LISTENING /tmp/.ICE-unix/4088
unix  2      [ ACC ] STREAM LISTENING /var/run/dbus/system_bus_socket
unix  3      [ ]     STREAM CONNECTED /var/run/dbus/system_bus_socket
unix  2      [ ]     DGRAM            @/var/run/hal/hotplug_socket
unix  2      [ ]     DGRAM            @udevd
unix  2      [ ACC ] STREAM LISTENING /tmp/xmms_ceres.0
unix  3      [ ]     STREAM CONNECTED /tmp/.X11-unix/X0
unix  3      [ ]     STREAM CONNECTED /tmp/.ICE-unix/4088
```

Example: Linux Netlink Sockets

Netlink Families

- NETLINK_ARPD - ARP table
- NETLINK_ROUTE - routing updates and modifications of IPv4 routing table
- NETLINK_ROUTE6 - routing updates and modifications of IPv6 routing table
- NETLINK_FIREWALL - IPv4 firewall
- ...

Network Subsystem Internals

Queuing Architecture

Example: Linux SK Buff Structure

SK_Buff Structure

```
struct sk_buff *alloc_skb (unsigned int size, int priority);
void skb_reserve (struct sk_buff *skb, unsigned int len);
int skb_headroom (const struct sk_buff *skb);
int skb_tailroom (const struct sk_buff *skb);

unsigned char *skb_put (struct sk_buff *skb, unsigned int len);
unsigned char *skb_push (struct sk_buff *skb, unsigned int len);

unsigned char *skb_pull (struct sk_buff *skb, unsigned int len);
void skb_trim (struct sk_buff *skb, unsigned int len);
```

Packet Filtering

Example: Linux Packet Filter

Network Filter Tables And Chains

The `filter` table is for normal packets:

- INPUT - chain for incoming packets
- OUTPUT - chain for outgoing packets
- FORWARD - chain for packets that pass through

The `nat` table is for packets that open new connections:

- PREROUTING
- OUTPUT
- POSTROUTING

The `mangle` table is for packets that need special modifications:

- PREROUTING
- INPUT
- OUTPUT
- FORWARD
- POSTROUTING

Network Filter Actions

The basic actions:

- ACCEPT - accept packet
- DROP - discard packet
- QUEUE - forward to application filter
- RETURN - continue previous chain

Examples of the extended actions:

- CONNMARK - mark connection associated with packet
- LOG - log packet and continue packet processing
- MARK - mark packet and continue packet processing
- MASQUERADE - enable address translation for connection
- REDIRECT - deliver packet locally
- REJECT - discard packet with notification
- ROUTE - apply given routing rules to packet
- TARPIT - hold connection associated with packet

Network Filters Example: Router

```
> cat /etc/sysconfig/iptables
*filter
:INPUT ACCEPT [0:0]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
:INPUT_FROM_LOCAL - [0:0]
:INPUT_FROM_WORLD - [0:0]
:FORWARD_FROM_LOCAL - [0:0]
:FORWARD_FROM_WORLD - [0:0]

# Sort traffic
-A INPUT -i lo -j INPUT_FROM_LOCAL
-A INPUT -i eth0 -j INPUT_FROM_LOCAL
-A INPUT -i tun0 -j INPUT_FROM_LOCAL
-A INPUT -i tun1 -j INPUT_FROM_LOCAL
-A INPUT -j INPUT_FROM_WORLD
-A FORWARD -i lo -j FORWARD_FROM_LOCAL
-A FORWARD -i eth0 -j FORWARD_FROM_LOCAL
-A FORWARD -i tun0 -j FORWARD_FROM_LOCAL
-A FORWARD -i tun1 -j FORWARD_FROM_LOCAL
-A FORWARD -j FORWARD_FROM_WORLD

# Input from local machines
-A INPUT_FROM_LOCAL -j ACCEPT

# Input from world machines
-A INPUT_FROM_WORLD -p tcp --dport ssh -j ACCEPT
-A INPUT_FROM_WORLD -p tcp --dport http -j ACCEPT
-A INPUT_FROM_WORLD -p tcp --dport smtp -j ACCEPT
-A INPUT_FROM_WORLD -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT_FROM_WORLD -j REJECT

# Forward from local machines
```

```
-A FORWARD_FROM_LOCAL -j ACCEPT

# Forward from world machines
-A FORWARD_FROM_WORLD -m state --state ESTABLISHED,RELATED -j ACCEPT
-A FORWARD_FROM_WORLD -j REJECT

COMMIT

*nat
:PREROUTING ACCEPT [0:0]
:POSTROUTING ACCEPT [0:0]
:OUTPUT ACCEPT [0:0]
-A PREROUTING -s 192.168.0.128/25 -p tcp --dport http -j REDIRECT --to-ports 3128
-A PREROUTING -s 192.168.0.128/25 -p tcp --dport smtp -j REDIRECT --to-ports 25
-A POSTROUTING -o ppp0 -s 192.168.0.128/25 -j MASQUERADE
COMMIT
```

Network Filters Example: Port Knocking

```
> cat /etc/sysconfig/iptables
*filter

...

# Rules to dispatch between port knock state machine states
# INSTEP1 to INSTEP3 are (arbitrary) names of address lists
-A INPUT -m recent --name INSTEP3 --rcheck -j STEP3
-A INPUT -m recent --name INSTEP2 --rcheck -j STEP2
-A INPUT -m recent --name INSTEP1 --rcheck -j STEP1
-A INPUT -j STEP0

# Machine state STEP0
# Jump to STEP1 if first knock comes, otherwise discard and stay in STEP0
-A STEP0 -p udp --dport 1111 -m recent --name INSTEP1 --set -j REJECT
-A STEP0 -j REJECT

# Machine state STEP1
# Jump to STEP2 if second knock comes, otherwise jump to STEP0
-A STEP1 -m recent --name INSTEP1 --remove
-A STEP1 -p udp --dport 2222 -m recent --name INSTEP2 --set -j REJECT
-A STEP1 -j STEP0

# Machine state STEP2
# Jump to STEP3 if third knock comes, otherwise jump to STEP0
-A STEP2 -m recent --name INSTEP2 --remove
-A STEP2 -p udp --dport 3333 -m recent --name INSTEP3 --set -j REJECT
-A STEP2 -j STEP0

# Machine state STEP3
# Accept one SSH connection attempt, otherwise jump to STEP0
-A STEP3 -m recent --name INSTEP3 --remove
-A STEP3 -p tcp --dport ssh -j ACCEPT
-A STEP3 -j STEP0

COMMIT
```

Example adjusted from literature, see references.

Example: Linux Packet Scheduling

Network Schedulers Example

```
# Root qdisc is prio with 3 bands
tc qdisc add dev ppp0 root handle 1: prio bands 3

# Band 1 qdisc is sfq and filter is ICMP & SSH & DNS & outbound HTTP
tc qdisc add dev ppp0 parent 1:1 sfq perturb 16
tc filter add dev ppp0 parent 1: protocol ip prio 1 u32 match ip protocol 1 0xff flowid
tc filter add dev ppp0 parent 1: protocol ip prio 1 u32 match ip sport 22 0xffff flowid
tc filter add dev ppp0 parent 1: protocol ip prio 1 u32 match ip dport 22 0xffff flowid
tc filter add dev ppp0 parent 1: protocol ip prio 1 u32 match ip sport 53 0xffff flowid
tc filter add dev ppp0 parent 1: protocol ip prio 1 u32 match ip dport 53 0xffff flowid
tc filter add dev ppp0 parent 1: protocol ip prio 1 u32 match ip sport 80 0xffff flowid

# Band 2 qdisc is sfq and filter is anything unfiltered
tc qdisc add dev ppp0 parent 1:2 sfq perturb 16
tc filter add dev ppp0 parent 1: protocol ip prio 9 u32 match u8 0 0 flowid 1:2

# Band 3 qdisc is tbf and filter is outbound SMTP
tc qdisc add dev ppp0 parent 1:3 tbf rate 128kbit buffer 100000 latency 100s
tc filter add dev ppp0 parent 1: protocol ip prio 1 u32 match ip dport 25 0xffff flowid
```

Network Subsystem Applications

File Systems

Example: Network File System

Basic Protocol Structures

```
const MNTPATHLEN = 1024;  /* maximum bytes in a pathname argument */
const MNTNAMLEN = 255;    /* maximum bytes in a name argument */
const FHSIZE = 32;        /* size in bytes of a file handle */

typedef opaque fhandle [FHSIZE];
typedef string name <MNTNAMLEN>;
typedef string dirpath <MNTPATHLEN>;

union fhstatus switch (unsigned fhs_status) {
  case 0:
    fhandle fhs_fhandle;
  default:
    void;
};
```

Mount Protocol Interface

```
typedef struct mountbody *mountlist;
struct mountbody {
  name ml_hostname;
  dirpath ml_directory;
  mountlist ml_next;
};
```

```
typedef struct groupnode *groups;
struct groupnode {
  name gr_name;
  groups gr_next;
};

typedef struct exportnode *exports;
struct exportnode {
  dirpath ex_dir;
  groups ex_groups;
  exports ex_next;
};

program MOUNTPROG {
  version MOUNTVERS {
    void MOUNTPROC_NULL (void) = 0;
    fhstatus MOUNTPROC_MNT (dirpath) = 1;
    mountlist MOUNTPROC_DUMP (void) = 2;
    void MOUNTPROC_UMNT (dirpath) = 3;
    void MOUNTPROC_UMNTALL (void) = 4;
    exports MOUNTPROC_EXPORT (void) = 5;
    exports MOUNTPROC_EXPORTALL (void) = 6;
  } = 1;
} = 100005;
```

NFS Protocol LOOKUP Interface Function

```
program NFS_PROGRAM {
  version NFS_VERSION {
    ...
    diropres NFSPROC_LOOKUP (diropargs) = 4;
    ...
  } = 2;
} = 100003;

struct diropargs {
  nfs_fh  dir;            /* directory file handle */
  filename name;          /* file name */
};

union diropres switch (nfsstat status) {
case NFS_OK:
  diropokres diropres;
default:
  void;
};

struct diropokres {
  nfs_fh file;
  fattr attributes;
};

struct fattr {
  ftype type;             /* file type */
  unsigned mode;          /* protection mode bits */
  unsigned nlink;         /* number of hard links */
  unsigned uid;           /* owner user id */
  unsigned gid;           /* owner group id */
  unsigned size;          /* file size in bytes */
  unsigned blocksize;     /* preferred block size */
  unsigned rdev;          /* special device number */
  unsigned blocks;        /* used size in kilobytes */
  unsigned fsid;          /* device number */
```

```
  unsigned fileid;        /* inode number */
  nfstime atime;          /* time of last access */
  nfstime mtime;          /* time of last modification */
  nfstime ctime;          /* time of last change */
};

struct nfs_fh {
  opaque data [NFS_FHSIZE];
};

enum nfsstat {
  NFS_OK=0,               /* No error */
  NFSERR_PERM=1,          /* Not owner */
  NFSERR_NOENT=2,         /* No such file or directory */
  NFSERR_IO=5,            /* I/O error */
  NFSERR_NXIO=6,          /* No such device or address */
  NFSERR_ACCES=13,        /* Permission denied */
  NFSERR_EXIST=17,        /* File exists */
  NFSERR_NODEV=19,        /* No such device */
  NFSERR_NOTDIR=20,       /* Not a directory*/
  NFSERR_ISDIR=21,        /* Is a directory */
  NFSERR_FBIG=27,         /* File too large */
  NFSERR_NOSPC=28,        /* No space left on device */
  NFSERR_ROFS=30,         /* Read-only file system */
  NFSERR_NAMETOOLONG=63,  /* File name too long */
  NFSERR_NOTEMPTY=66,     /* Directory not empty */
  NFSERR_DQUOT=69,        /* Disc quota exceeded */
  NFSERR_STALE=70,        /* Stale NFS file handle */
  NFSERR_WFLUSH=99        /* Write cache flushed */
};
```

NFS Protocol READ Interface Function

```
program NFS_PROGRAM {
  version NFS_VERSION {
    ...
    readres NFSPROC_READ (readargs) = 6;
    ...
  } = 2;
} = 100003;

struct readargs {
  nfs_fh file;            /* handle for file */
  unsigned offset;        /* byte offset in file */
  unsigned count;         /* immediate read count */
  unsigned totalcount;    /* total read count (from this offset)*/
};

union readres switch (nfsstat status) {
case NFS_OK:
  readokres reply;
default:
  void;
};

struct readokres {
  fattr attributes;       /* attributes */
  opaque data <NFS_MAXDATA>;
};

struct fattr {
  ftype type;             /* file type */
  unsigned mode;          /* protection mode bits */
  unsigned nlink;         /* number of hard links */
```

```
  unsigned uid;           /* owner user id */
  unsigned gid;           /* owner group id */
  unsigned size;          /* file size in bytes */
  unsigned blocksize;     /* preferred block size */
  unsigned rdev;          /* special device number */
  unsigned blocks;        /* used size in kilobytes */
  unsigned fsid;          /* device number */
  unsigned fileid;        /* inode number */
  nfstime atime;          /* time of last access */
  nfstime mtime;          /* time of last modification */
  nfstime ctime;          /* time of last change */
};

struct nfs_fh {
  opaque data [NFS_FHSIZE];
};

enum nfsstat {
  NFS_OK=0,               /* No error */
  NFSERR_PERM=1,          /* Not owner */
  NFSERR_NOENT=2,         /* No such file or directory */
  NFSERR_IO=5,            /* I/O error */
  NFSERR_NXIO=6,          /* No such device or address */
  NFSERR_ACCES=13,        /* Permission denied */
  NFSERR_EXIST=17,        /* File exists */
  NFSERR_NODEV=19,        /* No such device */
  NFSERR_NOTDIR=20,       /* Not a directory*/
  NFSERR_ISDIR=21,        /* Is a directory */
  NFSERR_FBIG=27,         /* File too large */
  NFSERR_NOSPC=28,        /* No space left on device */
  NFSERR_ROFS=30,         /* Read-only file system */
  NFSERR_NAMETOOLONG=63,  /* File name too long */
  NFSERR_NOTEMPTY=66,     /* Directory not empty */
  NFSERR_DQUOT=69,        /* Disc quota exceeded */
  NFSERR_STALE=70,        /* Stale NFS file handle */
  NFSERR_WFLUSH=99        /* Write cache flushed */
};
```

Chapter 7. Security Subsystem

Authentication

Linux PAM Example

PAM Functions

- Account management
- Authentication management
- Password management
- Session management

PAM Configuration Example

```
> cat /etc/pam.d/login
auth       required     pam_securetty.so
auth       required     pam_stack.so service=system-auth
auth       required     pam_nologin.so
account    required     pam_stack.so service=system-auth
password   required     pam_stack.so service=system-auth
session    required     pam_stack.so service=system-auth
session    optional     pam_console.so
```

PAM Usage Example

```
#include <security/pam_appl.h>
#include <security/pam_misc.h>

static struct pam_conv conv = { misc_conv, NULL };

int main(int argc, char *argv[])
{
  pam_handle_t *pamh = NULL;
  char *user;
  int retval;

  // ...

  retval = pam_start ("check_user", user, &conv, &pamh);
  if (retval == PAM_SUCCESS)
    retval = pam_authenticate (pamh, 0); // Is user really himself ?
  if (retval == PAM_SUCCESS)
    retval = pam_acct_mgmt (pamh, 0);  // Is user account valid ?
  if (retval == PAM_SUCCESS)

  // ...

  pam_end (pamh, retval);
}
```

Authorization

Example: Security Enhanced Linux

SELinux File Contexts

```
> ls -Z /
       system_u:object_r:bin_t:s0 bin
      system_u:object_r:boot_t:s0 boot
    system_u:object_r:device_t:s0 dev
       system_u:object_r:etc_t:s0 etc
 system_u:object_r:home_root_t:s0 home
       system_u:object_r:lib_t:s0 lib
       system_u:object_r:lib_t:s0 lib64
       system_u:object_r:mnt_t:s0 media
       system_u:object_r:mnt_t:s0 mnt
       system_u:object_r:usr_t:s0 opt
      system_u:object_r:proc_t:s0 proc
system_u:object_r:admin_home_t:s0 root
   system_u:object_r:var_run_t:s0 run
       system_u:object_r:bin_t:s0 sbin
       system_u:object_r:var_t:s0 srv
     system_u:object_r:sysfs_t:s0 sys
...
> semanage fcontext -l
SELinux fcontext        type               Context
/                       directory          system_u:object_r:root_t:s0
/.*                     all files          system_u:object_r:default_t:s0
/bin                    all files          system_u:object_r:bin_t:s0
/bin/.*                 all files          system_u:object_r:bin_t:s0
/bin/bash               regular file       system_u:object_r:shell_exec_t:s0
/bin/dmesg              regular file       system_u:object_r:dmesg_exec_t:s0
/bin/ip                 regular file       system_u:object_r:ifconfig_exec_t:s0
...
/dev                    directory          system_u:object_r:device_t:s0
/dev/.*                 all files          system_u:object_r:device_t:s0
/dev/.*mouse.*          character device   system_u:object_r:mouse_device_t:s0
/dev/[0-9].*            character device   system_u:object_r:usb_device_t:s0
/dev/[shmxv]d[^/]*      block device       system_u:object_r:fixed_disk_device_t:s0
...
/home                   directory          system_u:object_r:home_root_t:s0
/home/[^/]+             directory          unconfined_u:object_r:user_home_dir_t:s0
/home/[^/]+/www(/.+)?   all files          unconfined_u:object_r:httpd_user_content_t:s
...
```

SELinux Process Contexts

```
> ps -Z
LABEL                             PID TTY          TIME CMD
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 4891 pts/0 00:00:00 ps
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 5124 pts/0 00:00:00 bash
> id -Z
unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
```

SELinux Enforcement Rules

```
> semanage module -l
Module Name               Priority  Language
abrt                      100       pp
accountsd                 100       pp
acct                      100       pp
afs                       100       pp
aiccu                     100       pp
aide                      100       pp
ajaxterm                  100       pp
alsa                      100       pp
amanda                    100       pp
...
> sesearch -A -t sshd_key_t -p write
allow ssh_keygen_t sshd_key_t:file { append create getattr ioctl link lock open read re
allow sshd_keygen_t sshd_key_t:file { append create getattr ioctl link lock open read r
...
allow files_unconfined_type file_type:file { append audit_access create execute execute
...
allow ftpd_t non_security_file_type:file { append create getattr ioctl link lock open r
allow kernel_t non_security_file_type:file { append create getattr ioctl link lock open
...
allow sysadm_t non_security_file_type:file { append create getattr ioctl link lock open
...
```

SELinux Booleans

```
> getsebool -a
antivirus_can_scan_system --> off
antivirus_use_jit --> off
...
daemons_dump_core --> off
daemons_enable_cluster_mode --> off
daemons_use_tcp_wrapper --> off
daemons_use_tty --> off
...
ftpd_anon_write --> off
ftpd_full_access --> off
ftpd_use_nfs --> off
...
git_cgi_enable_homedirs --> off
git_cgi_use_nfs --> off
...
httpd_anon_write --> off
httpd_builtin_scripting --> on
httpd_can_check_spam --> off
httpd_can_connect_ftp --> off
httpd_can_network_connect --> off
httpd_can_network_memcache --> off
httpd_can_sendmail --> off
httpd_enable_cgi --> on
httpd_enable_homedirs --> off
httpd_use_nfs --> off
...
```

SELinux Audit Log

```
> tail /var/log/audit/audit.log
type=AVC msg=audit(1515657259.550:620585): avc:  denied  { open } for  pid=8358 comm="s
...
> audit2allow < /var/log/audit/audit.log
#============= nagios_t ==============
allow nagios_t initrc_var_run_t:file open;
...
> ls -Z /run/utmp
system_u:object_r:initrc_var_run_t:s0 /run/utmp
```

SELinux Policy Sources

```
policy_module(ssh, 2.4.2)

gen_tunable(allow_ssh_keysign, false)
gen_tunable(ssh_sysadm_login, false)

attribute ssh_server;
attribute ssh_agent_type;

type ssh_t;
type ssh_exec_t;
type ssh_home_t;
type sshd_exec_t;
...

allow ssh_t self:capability { setuid setgid ... };
allow ssh_t self:tcp_socket create_stream_socket_perms;
allow ssh_t self:unix_dgram_socket { create_socket_perms sendto };
allow ssh_t self:unix_stream_socket { create_stream_socket_perms connectto };
...

allow ssh_t sshd_key_t:file read_file_perms;
allow ssh_t sshd_tmp_t:dir manage_dir_perms;
allow ssh_t sshd_tmp_t:file manage_file_perms;
...

tunable_policy (`allow_ssh_keysign',`
    domain_auto_trans (ssh_t, ssh_keysign_exec_t, ssh_keysign_t)
    allow ssh_keysign_t ssh_t:fd use;
    allow ssh_keysign_t ssh_t:process sigchld;
    allow ssh_keysign_t ssh_t:fifo_file rw_file_perms;
')
...
```