Solaris Internals

Kernel Architecture & Implementation

Richard McDougall
Senior Staff Engineer - Performance & Availability Engineering
Sun Microsystems, Inc.
12 Network Circle, Menlo Park, Ca. 94025
richard.mcdougall@Eng.Sun.COM

Jim Mauro
Senior Staff Engineer - Performance & Availability Engineering
Sun Microsystems, Inc.
400 Atrium Drive, Somerset, NJ 08812
james.mauro@Eng.Sun.COM

About the instructors:

Richard McDougall is a Senior Staff Engineer in the Performance Availability Engineering group at Sun Microsystems, Inc., where he focuses on large systems performance and architecture. Richard has developed several tools for measurement, monitoring and sizing of UNIX systems, and has made several design enhancements to the SunOS kernel in the areas of memory management and file system I/O.

James Mauro is a Senior Staff Engineer in the Performance Availability Engineering group at Sun Microsystems, Inc., where he focuses on Solaris application performance, resource management and system recovery and availability.

Richard and James authored **Solaris Internals: Core Kernel Architecture**. Prentice Hall, ISBN 0-13-022496-0.

Richard can be reached at richard.mcdougall@eng.sun.com
James can be reached at james.mauro@eng.sun.com

Agenda

- **Goals, Non-Goals & Assumptions**
- **Introduction**
- **Kernel Features, Organization & Packages**
- **Kernel Services**
- **The Multithreaded Process Model**
- **Scheduling Classes & The Kernel Dispatcher**
- **Memory Architecture & Virtual Memory**
- **Files & File Systems**

Goals, Non-Goals & Assumptions

- **Goals**
  - Provide an architectural overview of the Solaris kernel
  - Discuss the major data structures and internal algorithms
  - Provide insight as to the practical application of the subject matter
- **Non-goals**
  - Solaris kernel development
  - How to develop and integrate device drivers, file systems, system calls and STREAMS modules
  - Device driver, STREAMS and TCP/IP Internals
- **Assumptions**
  - General familiarity with UNIX systems.
  - General familiarity with operating system concepts
  - General familiarity with the Solaris operating environment

Introduction

Introduction

- **What is Solaris?**

SOE - Solaris Operating Environment

3 major components:

- SunOS - the kernel (the 5.X thing)
- Windowing - desktop environment. CDE default, OpenWindows still included

  GNOME forthcoming
- Open Network Computing (ONC+). NFS (V2 & V3), NIS/NIS+, RPC/XDR

Solaris Distribution

- **12 CDs in the distribution**
  - WEB start CD (Installation)
  - OS bits, disks 1 and 2
  - Documentation (Answerbook)
  - Software Supplement (more optional bits)
  - Flash PROM Update
  - Maintenance Update
  - Sun Management Center
  - Forte’ Workshop (try n’ buy)
- **Bonus Software**
  - Software Companion (GNU, etc)
  - StarOffice (5.2a)
  - iPlanet Advantage Software (2 CDs)
  - Oracle 8i Enterprise Server (8.1.7)

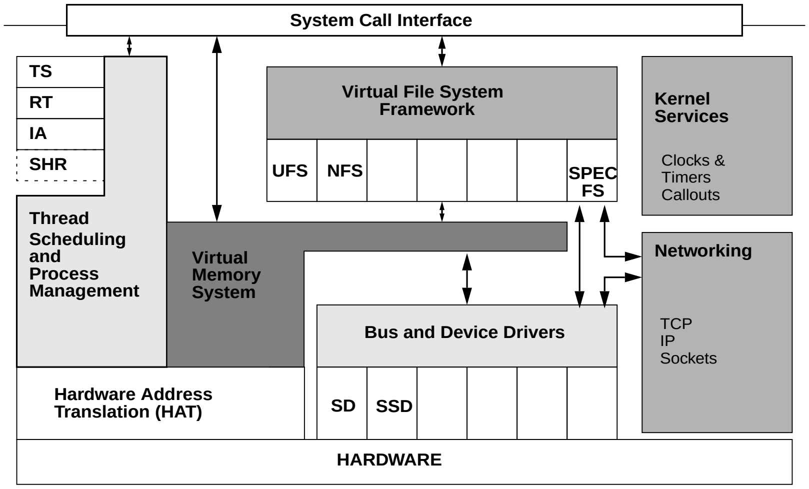
Releases

- **Base release, followed by quarterly update releases**
  - Solaris 8 - released 2/00
  - Solaris 8, 6/00 (update 1)
  - Solaris 8, 10/00 (update 2)
  - Solaris 8, 1/01 (update 3)
  - Solaris 8, 4/01 (update 4)

```
sunsys> cat /etc/release
                        Solaris 8 6/00 s28s_u1wos_08 SPARC
           Copyright 2000 Sun Microsystems, Inc.  All Rights Reserved.
                             Assembled 26 April 2000
sunsys>
```

Solaris Kernel Features & Organization

System Overview



Solaris Kernel Features

- **Dynamic Kernel**
  - Core unix/genunix modules
  - Major subsystems implemented as dynamically loadable modules (file systems, scheduling classes, STREAMS modules, system calls).
  - Dynamic resource sizing & allocation (processes, files, locks, memory, etc)
  - Dynamic sizing based on system size
    - Goal is to minimize/elminate need to use /etc/system tuneable parameters

Solaris Kernel Features

- **Preemptive kernel**
  - Does **NOT** require interrupt disable/blocking via PIL for synchronization
  - Most kernel code paths are preemptable
  - A few non-preemption points in critical code paths
  - SCALABILITY & LOW LATENCY INTERRUPTS
- **Well-defined, layered interfaces**
  - Module support, synchronization primitives, etc

Solaris Kernel Features

- **Multithreaded kernel**
  - Kernel threads perform core system services
  - Fine grained locking for concurrency
  - Threaded subsystems
- **Multithreaded process model**
  - User level threads and synchronization primitives
  - Solaris & POSIX threads
  - Two-level model isolates user threads from kernel

Solaris Kernel Features

- **Table-driven dispatcher with multiple scheduling class support**
  - Dynamically loadable/modifyable table values
- **Realtime support with preemptive kernel**
  - Additional kernel support for realtime applications (memory page locking, asynchronous I/O, processor sets, interrupt control, high-res clock)
- **Kernel tuning via text file (`/etc/system`)**
  - Some things can be done “on the fly”

    `adb`(1) `mdb`(1)

Solaris Kernel Features

- **Tightly integrated virtual memory and file system support**
  - Dynamic page cache memory implementation
- **Virtual File System (VFS) Implementation**
  - Object-like abstractions for files and file systems
  - Facilitates new features/functionality

    Kernel sockets via sockfs, /proc enhancements (procfs), Doors (doorfs), fdfs, swapfs, tmpfs
  - Disk-based, distributed & pseudo file systems

Solaris Kernel Features

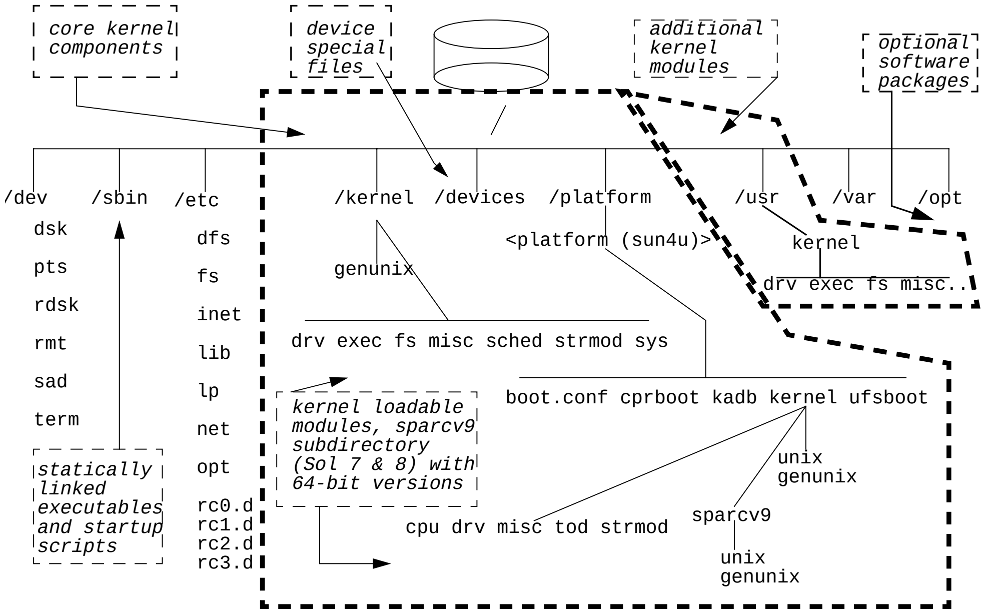
- **32-bit and 64-bit kernel**
  - 64-bit kernel *required* for UltraSPARC-III based systems (SunBlade, SunFire)
  - 32-bit apps run just fine...
- **Solaris DDI/DKI Implementation**
  - Device driver interfaces
  - Includes interfaces for dynamic attach/detach/pwr
- **Rich set of standards-compliant interfaces**
  - POSIX, UNIX International

Solaris Kernel Features

- **Integrated networking facilities**
  - TCP/IP

    IPv4, IPSec, IPv6
  - Name services - DNS, NIS, NIS+, LDAP
  - NFS - defacto standard distributed file system, NFS-V2 & NFS-V3
  - Remote Procedure Call/External Data Representation (RPC/XDR) facilities
  - Sockets, TLI, Federated Naming APIs

Kernel Organization



core kernel components

device special files

additional kernel modules

optional software packages

/

/dev /sbin /etc /kernel /devices /platform /usr /var /opt

dsk pts rdsk rmt sad term

dfs fs inet lib lp net opt rc0.d rc1.d rc2.d rc3.d

genunix

drv exec fs misc sched strmod sys

<platform (sun4u)>

boot.conf cprboot kadb kernel ufsboot

unix genunix

sparcv9

unix genunix

cpu drv misc tod strmod

kernel

drv exec fs misc..

kernel loadable modules, sparcv9 subdirectory (Sol 7 & 8) with 64-bit versions

statically linked executables and startup scripts

Solaris 8 Directory Namespace

- **A simple rule providing for the support and co-existence of 32-bit binaries on a 64-bit Solaris 8 system;**

  For every directory on the system that contains binary object files (executables, shared object libraries, etc), there is a `sparcv9` subdirectory containing the 64-bit versions

- ***All* kernel modules *must* be the of the same data model; ILP32 (32-bit data model) or LP64 (64-bit data model)**

- **64-bit kernel required to run 64-bit apps**

Solaris 8 Data Model

- **Defines the width of integral data types**
  - 32-bit Solaris - ILP32
  - 64-bit Solaris - LP64

’C’ data type	ILP32	LP64
char	8	8
short	16	16
int	32	32
long	**32**	**64**
longlong	64	64
pointer	**32**	**64**
enum	32	32
float	32	32
double	64	64
quad	128	128

Which Data Model Is Booted?

- **Use isainfo(1)**

```
sunsys> isainfo
sparcv9 sparc
sunsys> isainfo -v
64-bit sparcv9 applications
32-bit sparc applications
sunsys> isainfo -vk
64-bit sparcv9 kernel modules
```

- **Or isalist(1)**

```
sunsys> isalist -v
sparcv9+vis  sparcv9  sparcv8plus+vis  sparcv8plus
sparcv8 sparcv8-fsmuld sparcv7 sparc
```

- **man isaexec(3C)**

Solaris 8 Features

- **Kernel**
  - System error messages (Message IDs)
  - Virtual Memory Allocator (vmem)
  - Cyclics - arbitrary resolution timers
  - Remote console
  - /dev/poll driver
  - mmap(...,MAP_ANON,...), madvise(...,MADV_FREE,...)
  - Dynamic Reconfiguration
  - Alternate threads library (/usr/lib/lwp)

Solaris 8 Features (cont)

- **File Systems**
  - Forced unmount
  - UFS
    - deferred access time
    - logging (sol 7)
    - noatime (Sol 7)
    - directio concurrency
  - xmemfs
  - In-kernel mnttab (mntfs)
  - NFS Server Logging

Solaris 8 Features (cont)

- **Utilities**
  - `pkill`(1), `pgrep`(1) (Solaris 7)
  - `prstat`(1)
  - /proc tool improvements (`pstack, pmap, pldd, pcred` & `pflags` work on core files)
  - `dumpadm`(1M) (Solaris 7)
  - `coreadm`(1M)
  - Perl 5.005_03 bundled (YES!)
  - `apptrace`(1)
  - `vmstat`(1) paging statistics

Solaris 8 Features (cont)

- **Utilities (cont)**
  - `sort`(1) - much faster
  - `mdb`(1) - new modular debugger
  - `cpustat`(1) & `cputrack`(1)
  - `kstat`(1)
  - `lockstat`(1M)

    Does kernel profiling, as well as lock statistics

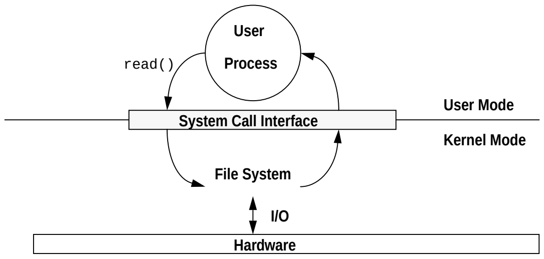
Kernel Services

Kernel Services

- **Traps & Interrupts**
- **System Calls**
- **System Clocks**
- **Kernel Callout Table**
- **Synchronization Primitives**
    - Mutex Locks
    - Dispatcher Locks
    - Reader/Writer Locks
    - Semaphores

Kernel Services

- **User processes/applications access kernel services through the *system call* facility**



- **Modes of execution (**kernel & user**) provide protection**
- **The kernel is entered through traps and interrupts**

Traps

- **A trap is a vectored transfer of control to specific kernel software designed to handle the trap**
- **A trap is one of many different types of events that can occurr while a CPU is executing instructions;**

  Resets
  MMU traps (page faults, etc)
  Register Window Exceptions
  Interrupts
  System calls

- **The kernel maintains a trap table (array of trap handlers) the base address of which is stored in a hardware register - Trap Base Address Register**

Traps

- **In SunOS parlance, the trap table is also called the System Control Block (SCB)**

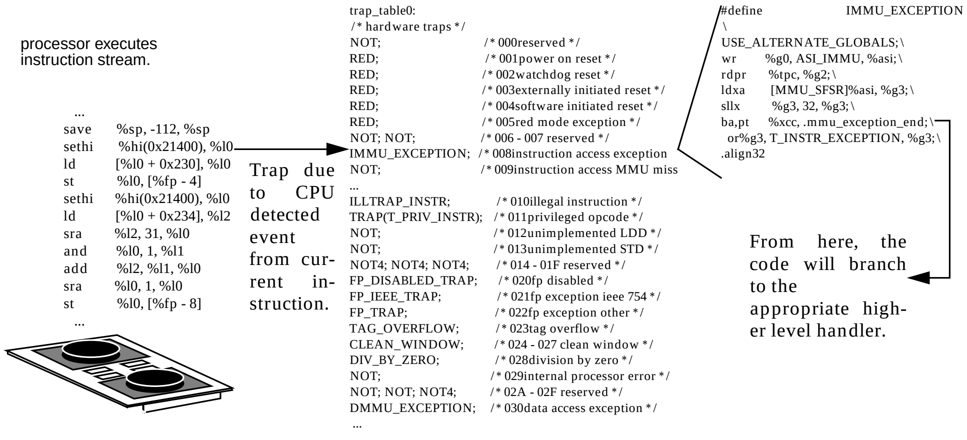
Table 1: Trap Table or SCB

Trap Description	Type	Priority
watchdog reset	002	1
instruction access MMU miss	009	2
illegal instruction	010	7
floating point exception	021	11
data access protection	033	12
divide by zero	028	15
trap instruction (e.g syscall is 108)	100 - 17f	16
interrupt level n (1 - 15)	041 - 04f	32-n

Partial trap table shown from sparcv9 (sun4u)

- **Trap table is hardware architecture specific**
- **Trap table is entered based on trap type and trap level**

Traps



processor executes instruction stream.

```
   ...
save      %sp, -112, %sp
sethi     %hi(0x21400), %l0
ld        [%l0 + 0x230], %l0
st        %l0, [%fp - 4]
sethi     %hi(0x21400), %l0
ld        [%l0 + 0x234], %l2
sra       %l2, 31, %l0
and       %l0, 1, %l1
add       %l2, %l1, %l0
sra       %l0, 1, %l0
st        %l0, [%fp - 8]
   ...
```

Trap due to CPU detected event from current instruction.

```
trap_table0:
 /* hardware traps */
NOT;                          /* 000reserved */
RED;                          /* 001power on reset */
RED;                         /* 002watchdog reset */
RED;                         /* 003externally initiated reset */
RED;                         /* 004software initiated reset */
RED;                         /* 005red mode exception */
NOT; NOT;                    /* 006 - 007 reserved */
IMMU_EXCEPTION;  /* 008instruction access exception
NOT;                         /* 009instruction access MMU miss
...
ILLTRAP_INSTR;             /* 010illegal instruction */
TRAP(T_PRIV_INSTR);   /* 011privileged opcode */
NOT;                         /* 012unimplemented LDD */
NOT;                         /* 013unimplemented STD */
NOT4; NOT4; NOT4;        /* 014 - 01F reserved */
FP_DISABLED_TRAP;      /* 020fp disabled */
FP_IEEE_TRAP;              /* 021fp exception ieee 754 */
FP_TRAP;                     /* 022fp exception other */
TAG_OVERFLOW;           /* 023tag overflow */
CLEAN_WINDOW;         /* 024 - 027 clean window */
DIV_BY_ZERO;              /* 028division by zero */
NOT;                        /* 029internal processor error */
NOT; NOT; NOT4;         /* 02A - 02F reserved */
DMMU_EXCEPTION;    /* 030data access exception */
 ...
```

```
#define                    IMMU_EXCEPTION
\
USE_ALTERNATE_GLOBALS;\
wr        %g0, ASI_IMMU, %asi;\
rdpr      %tpc, %g2;\
ldxa      [MMU_SFSR]%asi, %g3;\
sllx      %g3, 32, %g3;\
ba,pt     %xcc, .mmu_exception_end;\
  or%g3, T_INSTR_EXCEPTION, %g3;\
.align32
```

From here, the code will branch to the appropriate higher level handler.

Traps

- **Typical trap processing**
  - Set trap level
  - Save existing state in TSTATE register (CCR, ASI, PSTATE, CWP, PC, nPC)
  - Set PSTATE to predefined state for trap handling (processor to kernel mode, disable interrupts, set to alternate global registers)
  - Transfer control via trap table
- **UltraSPARC defines multiple trap levels, and can deal with nested traps**

Traps

- **The handler in the kernel, entered via the trap table, determines what mode the processor was in when the trap occurred**
    - Traps taken in user mode may result in a signal being sent to the process, which typically has a disposition to terminate the process
    - Error traps in kernel mode may cause a system crash, due to an unrecoverable error

    BAD TRAP: cpu=%d, type=%d, ...

    - Other traps may simply require work for the kernel, e.g. page faults start out as traps

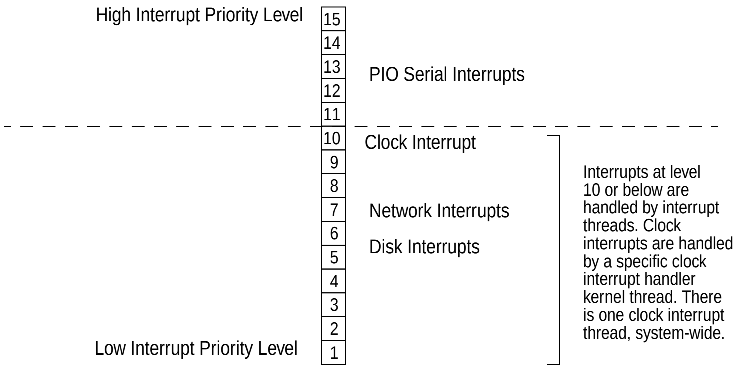
Interrupts

- **An asynchronous event, not associated with the currently executing instruction**
- **Like traps, interrupts result in a vectored transfer of control to a specific routine, e.g. a device interrupt handler (part of the device driver).**
- **Also like traps, interrupts are hardware architecture specific**
- **Interrupts can be “hard” or “soft”**
    - “Hard”ware interrupts generated by I/O devices
    - Soft interrupts are established via a call to the kernel `add_softintr()` function

Interrupts

- **Interrupt priority based on interrupt level; higher levels have higher priority**
- **The are 15 (1-15) interrupt levels defined**
  - Levels 1-9 are serviced by an interrupt thread linked to the processor that took the interrupt
  - Level 10 is the clock, and is handled by a dedicated *clock_intr_thread*
  - Levels 11-15 are handled in the context of the thread that was executing - these are considered high priority interrupts
  - Dispatcher locks are held at level 11

Interrupt Levels



High Interrupt Priority Level

15
14
13
12
11
10
9
8
7
6
5
4
3
2
1

Low Interrupt Priority Level

PIO Serial Interrupts

Clock Interrupt

Network Interrupts

Disk Interrupts

Interrupts at level 10 or below are handled by interrupt threads. Clock interrupts are handled by a specific clock interrupt handler kernel thread. There is one clock interrupt thread, system-wide.

Interrupt Levels

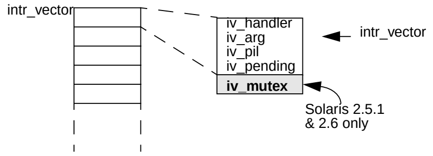
- **Typical system interrupt level assignments**

  Level 15 - Asynchronous memory errors
  Level 14 - Kernel profiling/deadman timer
  Level 13 - Cross calls (MP system xcall) & Audio device
  Level 12 - Console serial port
  Level 11 - Sbus level 6, Floppy controller
  Level 10 - Clock
  Levels 9 - 1, Devices, e.g. on-board SCSI level 4, frame buffer level 9, etc

- **Device interrupt levels can be gleaned from console out (/var/adm/messages)**

Interrupts

- **Interrupts are maskable by writing to the hardware Processor Interrupt Level (PIL) register**
  - Generic splx() kernel routines allow blocking of interrupts in critical sections
  - Interrupts => current PIL are allowed
- **UltraSPARC interrupt vector table**



Interrupts Levels

- **On UltraSPARC, there are a few interrupt levels that warrant symbolic representation;**

  CLOCK_LEVEL (10) - if you want to block the clock

  LOCK_LEVEL (10) - highest level you can be and still block

  DISP_LEVEL (11) - Must be at this level to run dispatcher functions

- **Interrupt PIL <= LOCK_LEVEL**
  - Handled by per-processor interrupt threads
  - Initialized and linked at boot time for each CPU

Interrupt Levels

- **Interrupt PIL == LOCK_LEVEL**
  - Essentially no different than other PIL 1-9 interrupts, except that the clock handler runs at level 10
  - Solaris 8 did away with the “clock_thread”
- **Interrupt PIL > LOCK_LEVEL**
  - High priority interrupts
  - Hijack the running kthread, and execute
  - No blocking allowed

Interrupts

- **Solaris implements an interrupt dispatch facility that sends an interrupt as a (small) packet of data to the target processor**
    - 24 bytes on US-I and US-II
    - 64-bytes on US-III
    - Commonly referred to as *mondo vectors*
- **Interrupt generation involves setting up several CPU registers, e.g. UltraSPARCs IDCR register**

```
/*
 * Interrupt Dispatch Command Register
 *            ASI_INTR_DISPATCH or ASI_SDB_INTR_W; ASI 0x77; VA = PORTID<<14|0x70
 *
 *            |---------------------------------------------------|
 *            |          0            | PORTID      |  0x70         |
 *            |----------------------|------------|---------------|
 *             63                    18         14  13            0
 */
#define IDCR_OFFSET0x70       /* IDCR VA<13:0> */
#define IDCR_PID_MASK0x7C000  /* IDCR VA<18:14> */
#define IDCR_PID_SHIFT14
```

Data-Bearing Mondo Vector

- **CPUs supply additional register space for interrupt data**
- **Solaris 8 added the DMV - Data-bearing Mondo Vector**
    - Take advantage of the register space to send more data along with the interrupt
    - New format for interrupt packets - unified to a single format for different interrupt types
    - Provides more efficient interrupt processing

Data-Bearing Mondo Vector

```
/*
 * DMV layout.
 *
 *                      +--+----------------+----------+---------------------------+
 *                      |63|62             61|60      48|47                       0|
 *                      +--+----------------+----------+---------------------------+
 *             Word 0:  | 1| reserved (MBZ)  | dmv_inum |     device private data     |
 *                      +--+----------+----------------+---------------------------+
 *             Word 1-7:| device private data                                     |
 *                      +----------------------------------------------------------+
 */
```

- **Bit 63 distinguishes a DMV from a conventional interrupt vector**
- **The kernel cross-call/cross-trap facility uses the DMV infrastructure to generate cross-calls and cross traps.**

Cross Calls (xcalls)

- **Xcalls are CPU-to-CPU interrupts, typically used for MMU related coherency tasks, CPU control, or forcing a CPU to enter the kernel**
- **They use the Mondo DMV facility to send interrupts, and can target a specific CPU, a group of CPUs, or all CPUs**
- **Two flavors; xcalls and xtraps**
    - xcalls execute at TL=0, interrupts enabled, PIL = 13
    - xtraps execute at TL > 0, interrupts disabled, PIL doesn’t matter

Interrupts

- **`intradm`(1M) - currently unbundled tool that allows for displaying and modifying interrupt bindings**

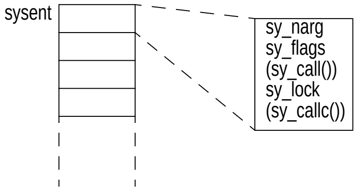
```
# /usr/bin/sparcv9/intradm
INUM  PIL                  DRIVER CPU PATH
  8b  5                   socal#0   1 /sbus@2,0/SUNW,socal@1,0
  95  9                   cgsix#0  11 /sbus@2,0/cgsix@2,0
  a2  5                     soc#0  10 /sbus@2,0/SUNW,soc@d,10000
  b9  c                     fhc#8  12 /central@1f,0/fhc@0,f8800000
  c3  5                   socal#1   0 /sbus@3,0/SUNW,socal@0,0
  db  5                     fas#0  12 /sbus@3,0/SUNW,fas@3,8800000
  dc  7                     hme#0  12 /sbus@3,0/SUNW,hme@3,8c00000
 18b  5                   socal#2  15 /sbus@6,0/SUNW,socal@1,0
 193  5                     soc#2   4 /sbus@6,0/SUNW,soc@2,0
 1a2  5                     soc#1   5 /sbus@6,0/SUNW,soc@d,10000
 1c3  5                   socal#3  14 /sbus@7,0/SUNW,socal@0,0
 1db  5                     fas#1  11 /sbus@7,0/SUNW,fas@3,8800000
 1dc  7                     hme#1  11 /sbus@7,0/SUNW,hme@3,8c00000
#
```

System Calls

- **A system call is a user-level process or thread requesting a service from the kernel**
- **System calls are documented in section 2 of the man pages, and are the core of the available APIs**
- **System calls are implemented via the aforementioned trap mechanism**
- `/etc/name_to_sysnum` **maps array entry to system call**

System Calls

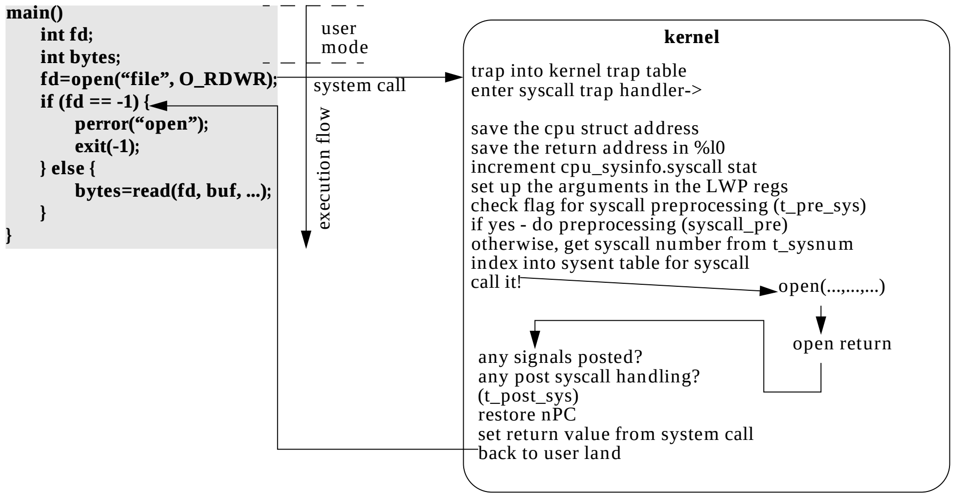
- The kernel maintains a system call entry table (sysent); 1 table entry for each system call
- Each table entry contains a sysent structure
- The table is indexed via the system call number



System Calls

- Some system calls are dynamically loadable kernel modules (e.g. Sys V IPC), others are loaded with the kernel during boot.
- New system calls can be added as dynamically loadable modules, which means you don’t need kernel source to do a kernel build to add a system call, but...
- You do need kernel source to code the system call properly
- `/etc/name_to_sysnum` is read at boot time to build the sysent table

System Calls



```
main()
    int fd;
    int bytes;
    fd=open("file", O_RDWR);
    if (fd == -1) {
        perror("open");
        exit(-1);
    } else {
        bytes=read(fd, buf, ...);
    }
}
```

trap into kernel trap table
enter syscall trap handler->

save the cpu struct address
save the return address in %l0
increment cpu_sysinfo.syscall stat
set up the arguments in the LWP regs
check flag for syscall preprocessing (t_pre_sys)
if yes - do preprocessing (syscall_pre)
otherwise, get syscall number from t_sysnum
index into sysent table for syscall
call it!

any signals posted?
any post syscall handling?
(t_post_sys)
restore nPC
set return value from system call
back to user land

System Calls

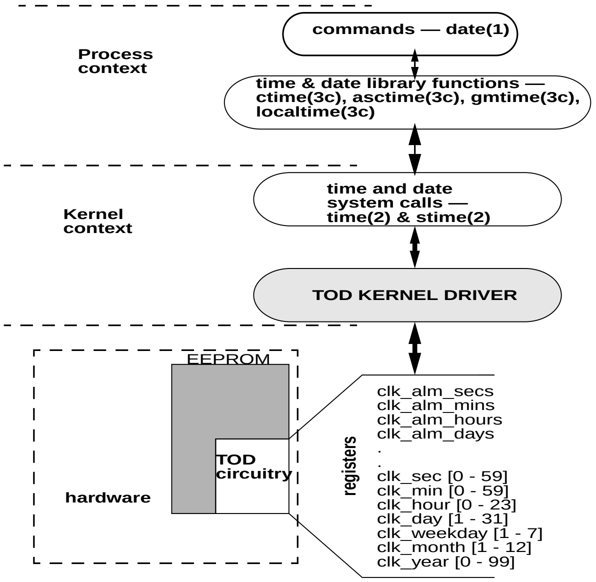
- **Kernel thread flags used in various places to flag required work**
  - `t_pre_sys`: pre-system call processing required, e.g. tracing, auditing, accounting
  - `t_post_sys, t_astflag, t_sigcheck`: post system call processing required

    profiling, signals, preemption
  - `t_sysnum`: number of the system call the kthread is currently executing (housekeeping)

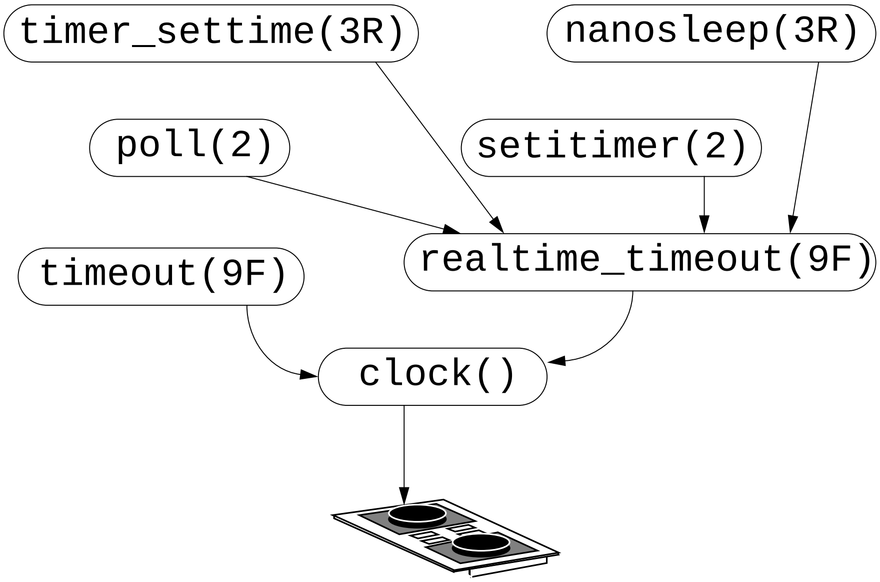
System Clocks

- All Sun systems implement a Time-Of-Day (TOD) clock chip that keeps time
- TOD clock circuitry is part of the system eeprom
- TOD device driver implemented to read/write TOD - accessable as a device
- Clock interrupts generated 100 times a second - every 10 milliseconds
- Clock interrupt handler performs generic housekeeping functions

System Clocks



Clocks & Timers



Cyclics

- **Solaris 8 introduced a new kernel subsystem that provides arbitrarily high-resolution, per-CPU interval timers; cyclics**
- **Designed to address short-comings in previous implementation**
  - Timeout resolution bound by clock frequency
  - Interval timers requiring re-priming the clock
  - Potential priority-inversion issues
- **Cyclics leverage modern microprocessor timer programmable registers (TICK, TICK_COMPARE on UltraSPARC)**

Cyclics

- **The subsystem provides callable interfaces by other kernel modules, a set of inter-cyclic interfaces, and a set of backend routines that are hardware architecture specific**
- **Linked list of cyclics off CPU structure**
- **Cyclics can fire at one of 3 interrupt levels; CY_LOW_LEVEL, CY_LOCK_LEVEL or CY_HIGH_LEVEL, specified by the caller when a cyclic is added.**

CY_LOCK_LEVEL == LOCK_LEVEL
CY_LOW_LEVEL must be < LOCK_LEVEL
CY_HIGH_LEVEL must be > LOCK_LEVEL

Cyclics

- **A cyclic client creates a client via the `cyclic_add()` kernel function, where the caller specifies;**
  - (function, arglist, level) and (absolute time since boot, and interval)
- **A CPU in the system partition is selected, the appropriate interrupt handler is installed, and the timers programmed.**
- **In Solaris 8, the `clock`() and `deadman`() functions are clients on the cyclic subsystem**

System Clocks

- **Clock interrupt handler**

```
Calculate free anon space
Calculate freemem
Calculate waitio
Calculate usr, sys & idle for each cpu
Do dispatcher tick processing
Increment lbolt
Check the callout queue
Update vminfo stats
Calculate runq and swapq sizes
Run fsflush if it’s time
Wake up the memory scheduler if necessary
```

System Clocks

- **Hardware watchdog timer**
  - Hardware clock in TOD circuit in EEPROM
  - Level 14 clock interrupt
  - Used for kernel profiling and deadman function
  - deadman must be explicitly enable (disabled by default)
  - deadman makes sure the level 10 clock is ticking. If it’s not, something is wrong, so save some state and call panic
  - Typically used to debug system hang problems
  - To enable deadman, set snooping in /etc/system & boot kadb (`set snooping = 1`)

Quick Tidbit

- **Look at lbolt if you’re not sure if the system is taking clock interrupts...**

```
# mdb -k
Loading modules: [ unix krtld genunix ip nfs ipc ptm logindmux ]
> lbolt/D
lbolt:
lbolt:          98136238
> lbolt/E
lbolt:
lbolt:          421495012254040063
> lbolt/E
lbolt:
lbolt:          421499633638850559
> lbolt/E
lbolt:
lbolt:          421501669453348863
> ::quit
#
```

Quick Tidbit

- **vmstat(1M) with the “`-i`” flag will do it also...**

```
# vmstat -i
interrupt         total     rate
--------------------------------
clock          27357130      100
zsc0                 10        0
zsc1            1701146        6
cgsixc0           19693        0
lec0                108        0
--------------------------------
Total          29078087      106
```

Quick Tidbit

- **Use `gethrtime(3C)` in code for fine grained measurement of functions (nanosecond granularity)**

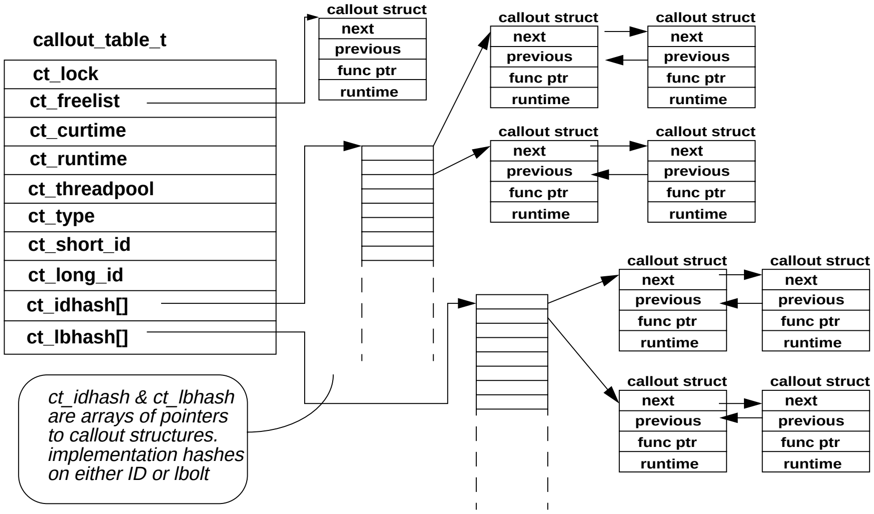
```
#include <time.h>
main()
{
          hrtime_t start, end;
          int i, iters = 100;

          start = gethrtime();
          for (i = 0; i < iters; i++)
               getpid();
          end = gethrtime();

          printf(“Avg getpid() time = %lld nsec\n”, (end - start) / iters);
}
sunsys> gt
Avg getpid() time = 2280 nsec
sunsys> gt
Avg getpid() time = 2270 nsec
sunsys> gt
Avg getpid() time = 2245 nsec
```

Kernel Callout Facility

- **Kernel callout facility is a method of providing general purpose event scheduling**
- **Enables the calling of a specific function at pre-determined time intervals**
- **Callout table initialized at boot time**
  - 2 Callout threads daemon created
- **Callout table populated via timeout(9F) kernel interface (Device Drivers)**

Kernel Callout Facility

- Kernel data structures for callout



ct_idhash & ct_lbhash are arrays of pointers to callout structures. implementation hashes on either ID or lbolt

Kernel Callout Facility

- **ct_lbhash contains “active” callout structs placed via timeout(9f)**
- **ct_idhash contains canceled timeout requests, from untimeout(9f)**
- **ct_threadpool is a condition variable used to nudge the callout_thread daemons**
- **Each callout structure contains a function pointer and arg pointer for the routine to get executed when the timer expires**

Quick Tidbit

- **mdb(1M) contains a function to dump the callout table;**

```
# mdb -k
Loading modules: [ unix krtld genunix ip nfs ipc ptm logindmux ]
> ::callout
FUNCTION                 ARGUMENT ID       TIME
realitexpire             70805520 7f746b31  5e77293 (T+249088)
polltime                 7015c188 3aeb5911  5e3a5c9 (T+54)
setrun                   402abe60 3aeb5951  5e3a671 (T+222)
mi_timer_fire            7087e6c8 7fe5c1d8  5e44955 (T+41922)
sigalarm2proc            70337580 7fe5a308  5e3f60e (T+20603)
ts_update                00000000 3dfab4a8  5e3a59a (T+7)
kmem_update              00000000 3dfab6f8  5e3a818 (T+645)
delay_wakeup             4023fe60 3dfab7c8  5e3bb4d (T+5562)
schedpaging              00000000 3b482ba9  5e3a5a8 (T+21)
qcallbwrapper            70355ea8 3b482c29  5e3a8f4 (T+865)
fas_watch                00000000 3b482c39  5e3a8f4 (T+865)
seg_pupdate              00000000 3b482cf9  5e3abdf (T+1612)
hme_check_link           70342000 3b482de9  5e3aee6 (T+2387)
.
.
.
>
```

Kernel CallBack Facility

- **Similiar to callout from an architecture standpoint**
- **Where callouts are time-driven, callbacks are event driven**
- **Currently used to support suspend/resume facility in various kernel segments**
  - e.g. VM system - Pages invalidated, faulted back in on resume

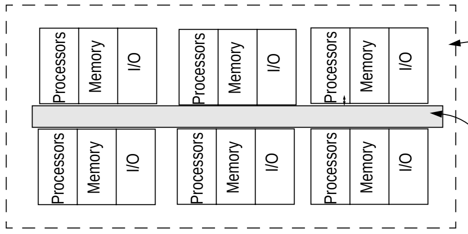
Synchronization Primitives

- **What are they and why do we need them?**

  Parallel Systems Architecture:

  - Multiprocessor systems with a single kernel image
  - SMP - shared memory multiprocessor, symmetric multiprocessor
  - Single, uniform address space

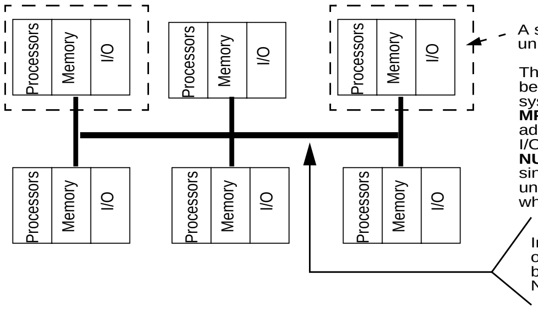
- **Need to synchronize access to kernel data with multiple processors executing kernel threads concurrently**



A single multiprocessor system.

shared memory, symmetric (SMP) system with uniform memory and I/O access. Single kernel image shared by all processors — single address space view.

System Interconnect — either bus or cross-bar design. Cache-coherent protocol for data transfers to/from memory, processors, and I/O. Very high, sustainable bandwidth with uniform access times (latency).



A single multiprocessor (or uniprocessor) node.

This hardware architecture could be an **MPP** or **NUMA/ccNUMA** system.
MPP — multiple OS images, multiple address space views, nonunifom I/O access.
NUMA/ccNUMA — single OS image, single addess space view, non-uniform memory, nonuniform I/O when interconnect is traversed.

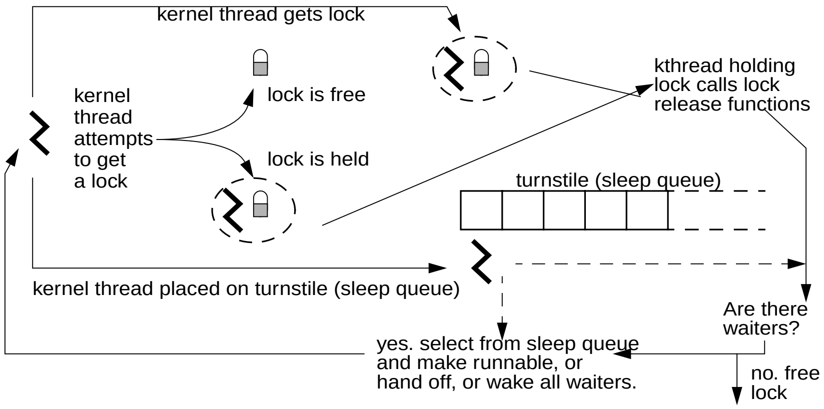
Interconnect is message-based on MPP platforms; memory-based, cache-coherent on NUMA/ccNUMA.

Synchronization Primitives

Solaris does NOT require manipulating PIL to block interrupts for most synchronization tasks...

- **Mutex (mutual exclusion) locks**
  - Most efficient - short hold times
- **Reader/Writer locks**
  - Allows mutiple readers, mutual exclusion semantics for writers (long hold times)
- **Semaphores**
  - Resource allocation

Lock Overview


kernel thread gets lock
kernel thread attempts to get a lock
lock is free
lock is held
kthread holding lock calls lock release functions
turnstile (sleep queue)
kernel thread placed on turnstile (sleep queue)
Are there waiters?
yes. select from sleep queue and make runnable, or hand off, or wake all waiters.
no. free lock

Mutex Locks

- **Lowest level, most efficient lock available**
- **There are basically 2 types of mutex locks;**
  - Adaptive mutex
  - Spin mutex
- **Adaptive is most frequently used - it's dynamic in what it does if the lock being sought after is held**
  - Is holder running? let's spin
  - Holder is not running, let's sleep

Mutex Locks

- **`lockstat`(1M)**
  - Implemented via /dev/lockstat pseudo device and driver
  - Provides for gathering/maintaining statistical information on kernel mutex and reader/writer locks
  - Also used for kernel profiling

    replaced kgmon(1M)

Reader/Writer Locks

- **Used when it’s OK to have multiple readers, but not OK to have multiple writers**
- **Implementation is a simple 64-bit word**

OWNER (writer) or HOLD COUNT (readers)	**wrlock**	**wrwant**	**wait**
63-3(LP64) 31-3(ILP32)	**2**	**1**	**0**

wait(0) indicates a thread is waiting for the lock. **wrwant(1)** indicates a writer wants the lock (prevents other readers from getting it). wrlock is the write lock.

wrlock(2) determines what the high bit will be; either the address of the writer thread, or the reader count.

Dispatcher Locks

- **Interrupts below level 10 can block, which means entering the dispatcher**
- **The dispatcher runs at PIL 11, in order to protect critical code paths from interrupts**
- **Dispatcher locks are synchronization primitives that not only provide mutual exclusion semantics, but also provide interrupt protection via PIL**

Semaphores

- **Traditionally could be used as binary (e.g. like a mutex) or counting (pool of resources)**
- **SunOS uses kernel semaphores in a few areas for resource allocation**

s_slpq
s_count

`s_slpq` `- pointer to linked list of kernel threads; the sleep queue for the semaphore`

`s_count` `- semaphore value`

Semaphores

- **Basic operations**

```
sema_p()
      if (count > 0)
         thread gets resource
      else
         put thread on sleep queue (s_slpq)
         swtch()

sema_v()
      count++
      if (s_slpq != NULL)
         wakeup highest priority waiting thread
```

Lock Statistics - lockstat

```
Adaptive mutex spin: 287 events

Count indv cuml rcnt     spin Lock                   Caller
------------------------------------------------------------------------------
  112  39%  39% 1.00      301 0x3000014d8e0          sdstrategy+0xac
   50  17%  56% 1.00        2 push_lock              queue_io_request+0x10
   22   8%  64% 1.00        1 push_lock              pageout+0x2c4
   19   7%  71% 1.00      244 0x3000014d8e0          sdintr+0x3c
   15   5%  76% 1.00       22 0x300003a6ee8          vmem_free+0x3c
   10   3%  79% 1.00        6 0x3000014d760          sdstart+0x53c
    8   3%  82% 1.00       12 0x300003a6ee8          vmem_xalloc+0xa4
    5   2%  84% 1.00       93 fhc_bdlist_mutex       fhc_bdlist_lock+0x8
    4   1%  85% 1.00        2 0x3000398f4a8          rdip+0x13c
    4   1%  87% 1.00       11 0x3000014d760          sdintr+0x3c
    4   1%  88% 1.00        1 0x30002c53e28          vn_rele+0x24
    3   1%  89% 1.00        5 0x3000014d760          sdstrategy+0xac
    3   1%  90% 1.00      815 0x3000014d8e0          sdstart+0x588
    3   1%  91% 1.00        1 0x300002061e0          isp_scsi_start+0x1f0
    2   1%  92% 1.00      675 0x3000014d8e0          sdstart+0x53c
    2   1%  93% 1.00       22 0x3000014d8e0          sdstrategy+0x2e0
    2   1%  93% 1.00    12401 pidlock                cv_wait_sig_swap+0x1b0
    2   1%  94% 1.00    20249 pidlock                exit+0x288
    2   1%  95% 1.00    25181 pidlock                lwp_exit+0x354
    1   0%  95% 1.00        8 cpc_mutex+0x50         page_list_add+0xec
    1   0%  95% 1.00     2526 pidlock                waitid+0xa8
    1   0%  96% 1.00      142 pidlock                sigcld_repost+0x48
    1   0%  96% 1.00        2 0x300002b6950          pm_idle_component+0xc
    1   0%  97% 1.00        2 ph_mutex+0x1a8         page_lookup+0x238
```

```
    1   0%  97% 1.00        2 pcf+0x108              page_free+0x128
    1   0%  97% 1.00       13 cpc_mutex+0x70         page_list_add+0xec
    1   0%  98% 1.00        2 ctr_mutex+0x50         page_ctr_add+0x38
    1   0%  98% 1.00        1 pse_mutex+0x360        page_trylock+0x20
    1   0%  98% 1.00        2 0x300002061e0          isp_scsi_start+0x164
    1   0%  99% 1.00        2 push_lock              cv_signal_pageout+0x1c
    1   0%  99% 1.00        2 push_lock              pageout+0x1c4
    1   0%  99% 1.00        2 cpc_mutex+0x60         page_get_mnode_cachelist+0xa4
    1   0% 100% 1.00        1 pcf+0x108              page_create_va+0x1a8
    1   0% 100% 1.00        2 cpc_mutex+0x20         page_list_add+0xec
------------------------------------------------------------------------------
```

Lock Statistics - lockstat

```
Adaptive mutex block: 3 events

Count indv cuml rcnt     nsec Lock                   Caller
-------------------------------------------------------------------------------
    2  67%  67% 1.00   107286 0x3000014d8e0          sdstrategy+0xac
    1  33% 100% 1.00    59704 0x3000014d8e0          sdintr+0x3c
-------------------------------------------------------------------------------
```

Lock Statistics - lockstat

```
Spin lock spin: 3314 events

Count indv cuml rcnt     spin Lock                   Caller
-------------------------------------------------------------------------------
 1399  42%  42% 1.00       27 cpu[8]+0x78            disp+0x94
  406  12%  54% 1.00       29 cpu[0]+0x78            disp+0x94
  296   9%  63% 1.00       33 cpu[9]+0x78            disp+0x94
  260   8%  71% 1.00       25 cpu[13]+0x78           disp+0x94
  254   8%  79% 1.00       26 0x30002bdf590          disp+0x94
  244   7%  86% 1.00       28 cpu[1]+0x78            disp+0x94
  153   5%  91% 1.00       21 cpu[12]+0x78           disp+0x94
  109   3%  94% 1.00       18 cpu[5]+0x78            disp+0x94
  103   3%  97% 1.00       27 cpu[7]+0x78            disp+0x94
   53   2%  99% 1.00      990 cpu[8]+0x78            disp_getbest+0xc
   35   1% 100% 1.00       49 cpu[6]+0x78            disp+0x94
    2   0% 100% 1.00      472 cpu[6]+0x78            disp_getbest+0xc
-------------------------------------------------------------------------------

Thread lock spin: 4 events

Count indv cuml rcnt     spin Lock                   Caller
-------------------------------------------------------------------------------
    1  25%  25% 1.00      211 cpu[6]+0x78            swapin+0x28
    1  25%  50% 1.00       86 cpu[0]+0x78            swapin+0x28
    1  25%  75% 1.00       56 sleepq_head+0xa08      ts_tick+0xc
    1  25% 100% 1.00       42 cpu[12]+0x78           swapin+0x28
-------------------------------------------------------------------------------
```

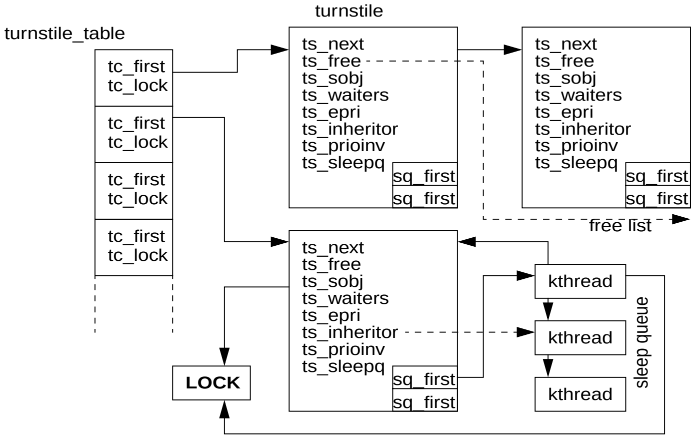
lockstat - kernel profiling

```
# lockstat -I sleep 20
Profiling interrupt: 3882 events in 20.011 seconds (194 events/sec)

Count indv cuml rcnt     nsec CPU+PIL              Caller
-------------------------------------------------------------------------------
  509  13%  13% 1.00      119 cpu[1]              i_ddi_splx+0x1c
  420  11%  24% 1.00      122 cpu[0]              i_ddi_splx+0x1c
  157   4%  28% 1.00       76 cpu[1]+10             spl6+0x14
  144   4%  32% 1.00       68 cpu[0]              disp_getwork+0x18
  142   4%  35% 1.00       70 cpu[0]              disp_getwork
  132   3%  39% 1.00       77 cpu[1]+10             i_ddi_splx
  116   3%  42% 1.00       81 cpu[1]              spl6
  115   3%  45% 1.00       72 cpu[0]+10             spl6+0x14
  115   3%  48% 1.00       72 cpu[0]+10             i_ddi_splx
  105   3%  50% 1.00       73 cpu[1]              disp_getwork
   96   2%  53% 1.00       64 cpu[0]              disp_getwork+0x10
   96   2%  55% 1.00       79 cpu[0]              spl6
   73   2%  57% 1.00       65 cpu[0]+10             disp_getwork+0x60
   71   2%  59% 1.00       69 cpu[1]              disp_getwork+0x18
   60   2%  61% 1.00       72 cpu[1]+10             disp_getwork+0x60
   60   2%  62% 1.00       67 cpu[1]              idle+0x74
   60   2%  64% 1.00       67 cpu[1]+10             disp_getwork+0x4c
```

Turnstiles and Priority Inheritance

- **Turnstile - A special set of sleep queues for kernel threads blocking on mutex or R/W locks**
- **Priority inheritance - a mechanism whereby a kernel thread may inherit the priority of the higher priority kernel thread, for the purpose of addressing;**
- **Priority inversion - a scenerio where a thread holding a lock is preventing a higher priority thread from running, because the higher priority thread needs the lock.**

Turnstiles and Priority Inheritance



Turnstiles

- **All active turnstiles reside in `turnstile_table`[], index via a hash function on the address of the synchronization object**
- **Each hash chain protected by a dispatcher lock, acquired by `turnstile_lookup`()**
- **Each kernel thread is created with a turnstile, in case it needs to block on a lock**
- **`turnstile_block`() - put the thread to sleep on the appropriate hash chain, and walk the chain, applying PI where needed**

Turnstiles

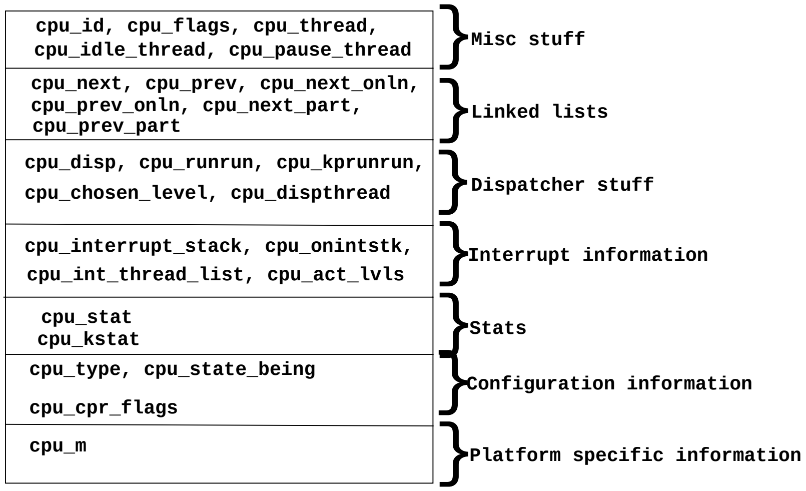
- **`turnstile_wakeup()` - waive an inherited priority, and wakeup the specific kernel threads**
- **For mutex locks, wakeup is called to wake all kernel threads blocking on the mutex**
- **For R/W locks;**
  - If no waiters, just release the lock
  - If a writer is releasing the lock, and there are waiting readers and writers, waiting readers get the lock if they are of the same or higher priority than the waiting writer
  - A reader releasing the lock gives priority to waiting writers

Kernel CPU Support

- **SunOS kernel maintains a linked list of CPU structures, one for each processor**
- **Facilitates many features, such as processor control (online/offline), processor binding, processor set**
- **Makes dispatcher implementation faster and more efficient**
- **Linked list gets created at boot time**

Kernel CPU Support

- **CPU data structure**



cpu_id, cpu_flags, cpu_thread, cpu_idle_thread, cpu_pause_thread	Misc stuff
cpu_next, cpu_prev, cpu_next_onln, cpu_prev_onln, cpu_next_part, cpu_prev_part	Linked lists
cpu_disp, cpu_runrun, cpu_kprunrun, cpu_chosen_level, cpu_dispthread	Dispatcher stuff
cpu_interrupt_stack, cpu_onintstk, cpu_int_thread_list, cpu_act_lvls	Interrupt information
cpu_stat cpu_kstat	Stats
cpu_type, cpu_state_being cpu_cpr_flags	Configuration information
cpu_m	Platform specific information

Kernel CPU Support

- **A CPU can be on any one of several linked lists**
  - Exists - all cpus
  - Online - all cpus that are online (not queisced)
  - Partition - part of a processor set
- **A CPU can be in one of several states (cpu_flags)**

```
CPU_RUNNING  - executing code
CPU_READY    - can accept cross-calls
CPU_QUIESCED - no threads
CPU_EXISTS   - it’s configured
CPU_ENABLE   - enabled for interrupts
CPU_OFFLINE  - no threads
CPU_POWEROFF - powered off
```

Kernel CPU Support

- **Each CPU holds a kthread pointer to the thread it’s currently executing, its idle thread and pause thread**
- **Each CPU has its own interrupt stack, and a linked list of 9 interrupt threads for handling interrupts below level 10**
- **CPU partitions are how processor sets are created and maintained (SunOS 5.6 and beyond). A `cpupart` structure is linked to the CPU structure**
- **Statistics include a `cpu_stat` structure, which is sysinfo, vminfo and wait info all merged together. kstats are available as well**

Quick Tidbit

- **There’s an adb macro for dumping a cpu structure**

```
# adb -k /dev/ksyms /dev/mem
physmem fdde
cpu_list/X
cpu_list:
cpu_list:       f026de48
f026de48$<cpu
cpus:
cpus:           id              seqid           flags
                0               0               1b
cpus+0xc:       thread          idle_t          pause
                f68181a0        fbe01e80        fbf53e80
cpus+0x18:      lwp             callo           fpowner         part
                f6438ca0        0               f6438ca0        f026f034
cpus+0x2c:      next            prev            next on         prev on
                f026e3b8        f026e3b8        f026e3b8        f026e3b8
cpus+0x3c:      next pt         prev pt
                f026e3b8        f026e3b8
cpus+0x44:      lock    npri    queue           limit           actmap
                0       110     f5b24008        f5b24530        f59810d0
cpus+0x54:      maxrunpri       max unb pri     nrunnable
                -1              -1              0
cpus+0x60:      runrun  kprnrn  chosen_level    dispthread
                0       0       -1              f68181a0
cpus+0x68:      thread lock     last_swtch      intr_stack      on_intr
                0               1b832b6         fbe1ffa0        0
cpus+0x78:      intr_thread     intr_actv       base_spl
                fbe1ce80        0               0
f026e3b8$<cpu
cpus+0x570:     id              seqid           flags
                2               1               1b
cpus+0x57c:     thread          idle_t          pause
                fbf58e80        fbf58e80        fbf78e80
cpus+0x588:     lwp             callo           fpowner         part
                f5c9e868        0               f6437440        f026f034
cpus+0x59c:     next            prev            next on         prev on
                f026de48        f026de48        f026de48        f026de48
```

```
cpus+0x5ac:     next pt         prev pt
                f026de48        f026de48
cpus+0x5b4:     lock    npri    queue           limit           actmap
                0       110     f5c64aa0        f5c64fc8        f5c02d10
cpus+0x5c4:     maxrunpri       max unb pri     nrunnable
                -1              -1              0
cpus+0x5d0:     runrun  kprnrn  chosen_level    dispthread
                0       0       -1              f5c67b20
cpus+0x5d8:     thread lock     last_swtch      intr_stack      on_intr
                0               1b842ee         fbf76fa0        0
cpus+0x5e8:     intr_thread     intr_actv       base_spl
                fbf73e80        0               0
f026f034$<cpupart
cp_default:
cp_default:
                id              level           next            prev
                0               0               f026f034        f026f034
cp_default+0x10:
                base            cpulist         ncpus
                f026f034        f026de48        2
cp_default+0x1c:                lock    npri    queue           limit           actmap

                              0       110     f5b24548        f5b24a70        f59810e0
cp_default+0x2c:                maxrunpri       max unb pri     nrunnable
                                    -1              -1              0
```

CPU Info

```
# mdb -k
Loading modules: [ unix krtld genunix ip nfs lofs ipc ptm logindmux ]
> ::cpuinfo
ID ADDR     FLG NRUN BSPL PRI RNRN KRNRN SWITCH THREAD           PROC
 0 1041add8  1b    5    0 104   no    no t-0    000002a10004bd40 sched
 1 02325528  1b    8    0  59   no    no t-0    0000030003d61aa0 oracle
 4 02324028  1b    6    0  59   no    no t-0    0000030007b8f260 oracle
 5 025d8ab0  1b   10    0  59   no    no t-0    0000030003d682e0 oracle
8 025cf538  2f    0    0  -1   no    no t-9621305 000002a100497d40 (idle)
 9 025ce038  2f    0    0  -1   no    no t-9621272 000002a10048bd40 (idle)
10 025ccac0  2f    0    0  -1   no    no t-7244620 000002a10053fd40 (idle)
11 025cb548  2f    0    0  -1   no    no t-7244620 000002a100533d40 (idle)
12 025ca048  2f    0    0  -1   no    no t-7244620 000002a100527d40 (idle)
13 025c6ad0  2f    0    0  -1   no    no t-7244619 000002a10063bd40 (idle)
14 025c3558  1b    7    0  59   no    no t-0    0000030007dbba60 mdb
15 025c2058  1b    8    0  59   no    no t--1   0000030003d68ac0 oracle
>
```

Processor Control Commands

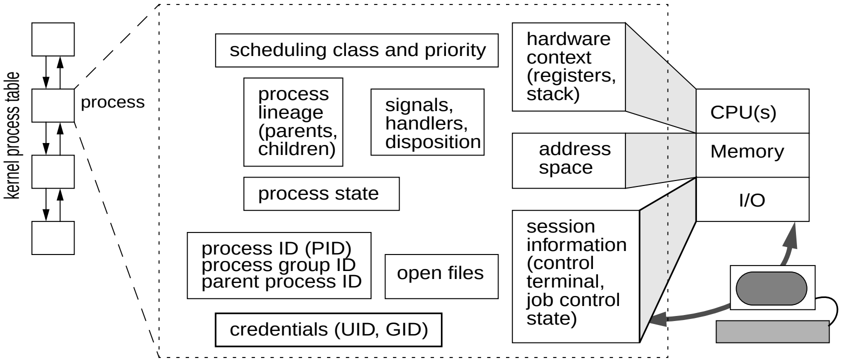
- **CPU related commands**
  - psrinfo(1M) - provides information about the processors on the system. Use “-v” for verbose
  - psradm(1M) - online/offline processors. Pre Sol 7, offline processors still handled interrupts. In Sol 7, you can disable interrupt participation as well
  - psrset(1M) - creation and management of processor sets
  - pbind(1M) - original processor bind command. Does not provide exclusive binding
  - processor_bind(2), processor_info(2), pset_bind(2), pset_info(2), pset_creat(2), p_online(2): system calls to do things programmatically

Processes, Threads and the Dispatcher

Processes, Threads & The Dispatcher

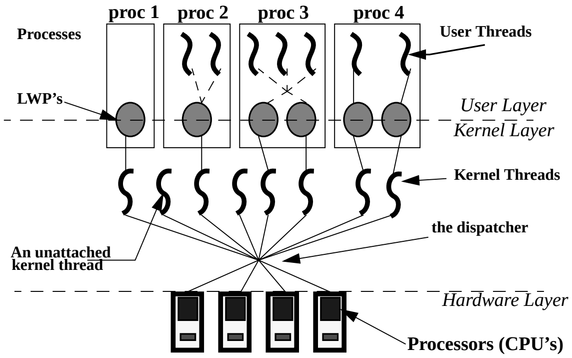
- **Solaris implements a multithreaded process model**
  - Traditional “proc” structure and user area (uarea)
- **New abstractions in the form of data structures**
  - Kernel Thread (kthread)
  - Lightweight Process (LWP)
- **Every process has at least one Kthread/LWP**
  - They always travel in pairs at user-process level
  - The inverse is not always true - kernel threads created by the OS do not have a corresponding LWP

Process Execution Environment


kernel process table
process
scheduling class and priority
process lineage (parents, children)
signals, handlers, disposition
process state
process ID (PID) process group ID parent process ID
open files
credentials (UID, GID)
hardware context (registers, stack)
address space
session information (control terminal, job control state)
CPU(s)
Memory
I/O

Multithreaded Process Model

- **Processes can have varying numbers of user threads, LWPs and kernel threads**



Kernel Process Model

- **So, what's a process?**

  "a process is the executable form of a program"

  [now that we got that out of the way...]

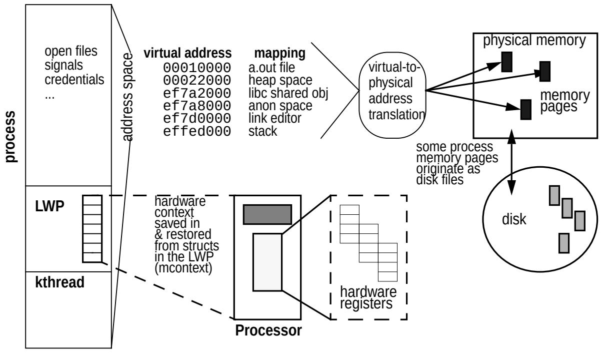
  - All processes begin life as a program
  - All processes begin life as a disk file (ELF object)
  - All processes have "state" or context that defines their execution environment
  - Context can be further divided into "hardware" context and "software" context

Kernel Process Model

- **Hardware context**
  - The processor state, which is CPU architecture dependent.
  - In general, the state of the hardware registers (general registers, privileged registers)
  - Maintained in the LWP
- **Software context**
  - Address space, credentials, open files, resource limits, etc - stuff shared by all the threads in a process

Kernel Process Model

- **The diagram below provides a “conceptual” view of process context**

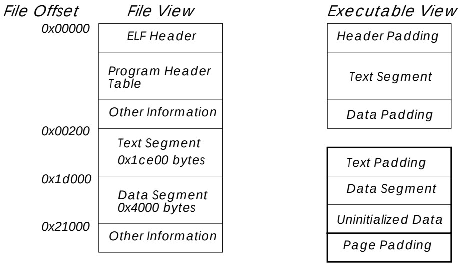


virtual address	**mapping**
00010000	a.out file
00022000	heap space
ef7a2000	libc shared obj
ef7a8000	anon space
ef7d0000	link editor
effed000	stack

process
open files
signals
credentials
...
address space
LWP
kthread
virtual-to-physical address translation
physical memory
memory pages
some process memory pages originate as disk files
disk
hardware context saved in & restored from structs in the LWP (mcontext)
Processor
hardware registers

Kernel Process Model

- **The Process Image defined by the System V & SPARC Application Binary Interface (ABI)**
- **2 part spec - platform dependent and platform independent**
- **Executable & Linking Format (ELF) object file spec**



File Offset *File View* *Executable View*

0x00000 ELF Header | Header Padding

Program Header Table | Text Segment

Other Information | Data Padding

0x00200 Text Segment 0x1ce00 bytes | Text Padding

0x1d000 Data Segment 0x4000 bytes | Data Segment

0x21000 Other Information | Uninitialized Data

Page Padding

Kernel Process Model

- **ELF provides the format definitions for the on-disk and in-ram formats**
- **ELF files divided into several well-defined sections**
  - ELF Header - describes the various components of the ELF file; a roadmap of the ELF file
  - Program Header Table (PHT) - array of Elf_Phdr data structures, each structure describes a segment of the ELF file for exec
  - Section Header Table (SHT) - array of Elf_Shdr structures, each structure describes a section of the ELF file

Kernel Process Model

- **ELF definition provides for ELF32 and ELF64 file formats**
  - Width of data types is different - data structure formats/contents are not changed
- **ELF sections**
  - ELF Header (sys/elf.h) - Generic information, such as file type, machine architecture, offsets into the PHT and SHT, etc
  - SHT - “pieces” of the ELF; symbol table, string table, symbol hash table, dynamic linking info, etc
  - PHT - Executables and shared objects only. Info needed for program execution; address, size, alignment, etc. Read by exec(2)

Kernel Process Model

- **ELF on-disk object created by the link-editor at the tail-end of the compilation process (although we still call it an a.out by default...)**
- **ELF objects can be *statically* linked or *dynamically* linked**
  - Compiler “-B static” flag, default is dynamic
  - Statically linked objects have all references resolved and bound in the binary (`libc.a`)
  - Dynamically linked objects rely on the run-time linker, ld.so.1, to resolve references to shared objects at run time (`libc.so.1`)
  - Static linking is discouraged, and not possible for 64-bit binaries in Solaris 7

Quick Tips

- **Use elfdump(1) to examine different pieces of an ELF file:**

```
fawlty> elfdump -e /bin/ls

ELF Header
  ei_magic:   { 0x7f, E, L, F }
  ei_class:   ELFCLASS32          ei_data:      ELFDATA2MSB
  e_machine:  EM_SPARC            e_version:    EV_CURRENT
  e_type:     ET_EXEC
  e_flags:                     0
  e_entry:               0x10f5c  e_ehsize:     52  e_shstrndx:   23
  e_shoff:                0x45dc  e_shentsize:  40  e_shnum:      24
  e_phoff:                  0x34  e_phentsize:  32  e_phnum:       5
fawlty>
```

  - Above is ELF header dump from /usr/bin/ls

Quick Tips

- **Section Header Table - -elfdump -c filename**

```
fawlty> elfdump -c /bin/ls

Section Header[9]:  sh_name: .text
    sh_addr:      0x10f5c         sh_flags:   [ SHF_ALLOC  SHF_EXECINSTR ]
    sh_size:      0x2b28          sh_type:    [ SHT_PROGBITS ]
    sh_offset:    0xf5c           sh_entsize: 0
    sh_link:      0               sh_info:    0
    sh_addralign: 0x4

Section Header[19]:  sh_name: .data
    sh_addr:      0x24348         sh_flags:   [ SHF_WRITE  SHF_ALLOC ]
    sh_size:      0x144           sh_type:    [ SHT_PROGBITS ]
    sh_offset:    0x4348          sh_entsize: 0
    sh_link:      0               sh_info:    0
    sh_addralign: 0x8

Section Header[21]:  sh_name: .bss
    sh_addr:      0x244f0         sh_flags:   [ SHF_WRITE  SHF_ALLOC ]
    sh_size:      0x7a4           sh_type:    [ SHT_NOBITS ]
    sh_offset:    0x44f0          sh_entsize: 0
    sh_link:      0               sh_info:    0
    sh_addralign: 0x8
```

Quick Tidbit

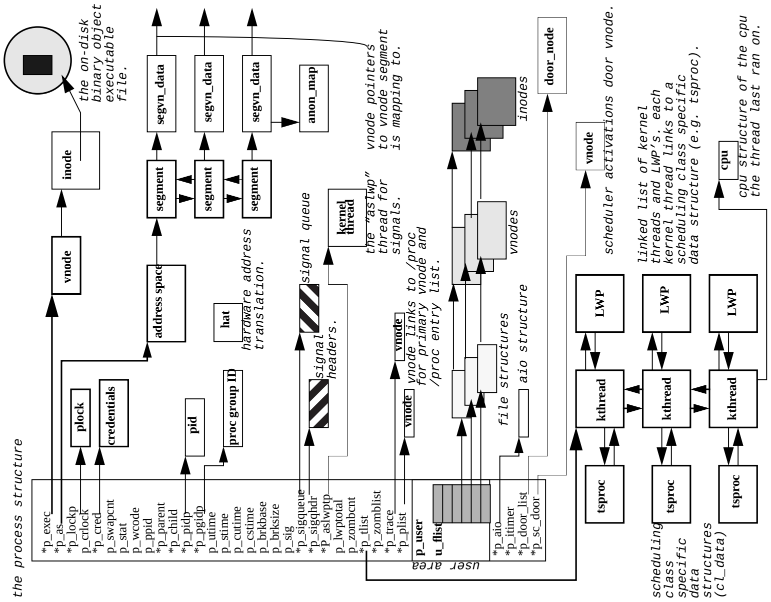
- **In addition to elfdump(1), there's pvs(1) and ldd(1)**
  - pvs(1) provides version information

```
fawlty> pvs  -r  /usr/bin/ldd  /usr/bin/pvs
/usr/bin/ldd:
        libelf.so.1 (SUNW_1.2);
        libc.so.1 (SUNW_1.1);
/usr/bin/pvs:
        libelf.so.1 (SUNW_1.2);
        libc.so.1 (SUNW_1.1);
fawlty>
```

  - ldd(1) provides dynamic linking information

```
fawlty> ldd /usr/bin/pvs
        libelf.so.1 =>   /usr/lib/libelf.so.1
        libc.so.1 =>     /usr/lib/libc.so.1
        libdl.so.1 =>    /usr/lib/libdl.so.1
fawlty>
```

The big picture...



Kernel Process Model

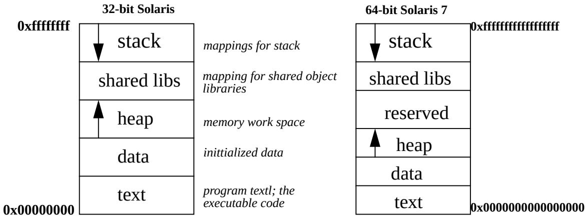
- **The proc structure (sys/proc.h) links to all the external structures that define the context and execution environment for the process**
  - Some things are imbedded with the proc struct; PID, PPID, state, counts, etc
  - Most stuff is defined via an external data structure, linked by a pointer in the proc structure; process lineage, address space, LWPs/kthreads, open files, scheduling class information, etc
  - User threads not shown

Kernel Process Model

- **Proc structure members**

`**p_exec** - points to vnode of exec'd object file`

`**p_as** - address space structure mappings`



`**p_cred** - credentials structure (IUD, eUID, etc)`

`**p_stat** - process state`

Kernel Process Model

- **Proc structure members (cont)**

p_pidp - PID structure pointer

p_ppid - parent PID

p_sessp - session structure pointer - process control terminal management

p_user - imbedded user structure (uarea)

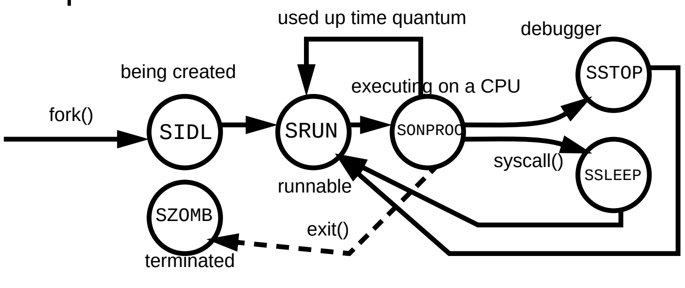
p_aio - asynchronous I/O structure pointer

p_model - SunOS 5.7 & 5.8 only, data model (ILP32 or LP64)

p_tlist - kthread pointer. Root of linked list of kthreads (if there’s more than one in the process)

Kernel Process Model

- **Process states**
  - Somewhat misleading - kthreads change state, not processes



  - For the most part, for each process state, there is a corresponding kthread state

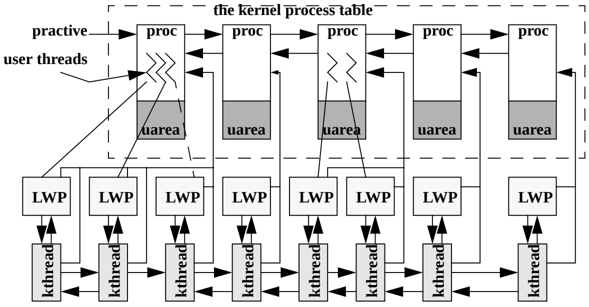
Kernel Process Model

Process States	Kthread States
SIDL	
SRUN	TS_RUN
SONPROC	TS_ONPROC
SSLEEP	TS_SLEEP
SSTOP	TS_STOPPED
SZOMB	TS_ZOMB
	TS_FREE

- Kthread creation is not flagged as a distinct state - they go right to TS_RUN
- Kthread structures are flagged as TS_FREE when the proc or kthread/LWP is terminated

Kernel Process Model

- **Kernel maintains system-wide linked lists of processes, LWPs and kthreads**



- **Relationship links maintained at every level**

Kernel Process Model

- **Kernel Process Table**
  - System-wide table of all process on the system
  - Max size based on `maxusers` (described) earlier
  - kmem cache allocator dynamically allocates space for new proc structures (it’s not a static array)
  - Look at `max_nprocs` or `v.v_procs` for max number
  - sar(1M) will also do the trick...

Kernel Process Model

- **Process table size**

```
# /etc/crash
dumpfile = /dev/mem, namelist = /dev/ksyms, outfile = stdout
> v
v_buf: 100
v_proc: 1914
v_nglobpris: 110
v_maxsyspri:  99
          .
          .
          .
v_bufhwm: 2456
> od max_nprocs
10413104:  0000077a
> od -d max_nprocs
10413104:  0000001914
> q
# sar -v 1 1

SunOS rascals 5.7 Generic sun4u    05/05/99

17:00:37  proc-sz    ov  inod-sz    ov  file-sz    ov   lock-sz
17:00:38   77/1914    0 3192/8452    0  563/563     0    0/0
```

Kernel Process Model

- **The user area, or uarea**
  - Traditional implementations of UNIX linked to uarea from proc structure

u_tsize, u_dsize	*text & data size*
u_start	*process start time*
u_psargs[], u_comm[]	*args to proc*
u_argc, u_argv, u_envp	*main(argc, argv, envp)*
u_cmask	*file creation mask*
u_rlimit[]	*array of resource limits*
u_nofiles, u_flist	*open files*
u_signal[]	*array of signal handlers*

  - Selected bits from the uarea above

Kernel Process Model

- **Process resource limits**
  - Maintained u_rlimits[] array of rlimits structure, where each structure defines a current and max value for a resource limit
  - Examined and changed via limit(1) or ulimit(1), or programmatically via setrlimit(2)/getrlimit(2)
  - SunOS 5.7 added the plimit(1) command, making things easier

```
CPU    - Max cpu time in milliseconds
FSIZE  - Max file size
DATA   - Max size of process data segment
STACK  - Max stack size
CORE   - Max core file size
NOFILE - Max number of open files
VMEM   - Max address space size
```

Kernel Process Model

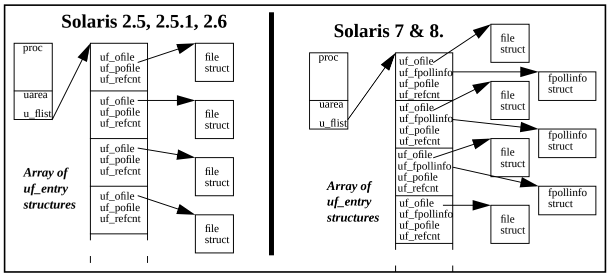
- **Resource limit defaults**

```
> p 62
PROC TABLE SIZE = 4058
SLOT ST  PID  PPID  PGID   SID   UID PRI   NAME        FLAGS
  62 s 24027   487 24027   487     0  55 sh             load
> u 62
PER PROCESS USER AREA FOR PROCESS 62
PROCESS MISC:
        command: sh, psargs: sh
        start: Wed May  5 22:45:36 1999
        mem: 6cc, type: exec su-user
        vnode of current directory: f6734f18
OPEN FILES, POFILE FLAGS, AND THREAD REFCNT:
        [0]: F 0xf64e64d8, 0, 0 [1]: F 0xf64e64d8, 0, 0
        [2]: F 0xf64e64d8, 0, 0
 cmask: 0022
RESOURCE LIMITS:
        cpu time: 18446744073709551613/18446744073709551613
        file size: 18446744073709551613/18446744073709551613
        swap size: 2147479552/18446744073709551613
        stack size: 8388608/2147479552
        coredump size: 18446744073709551613/18446744073709551613
        file descriptors: 64/1024
        address space: 18446744073709551613/18446744073709551613
```

  - Above from /etc/crash session

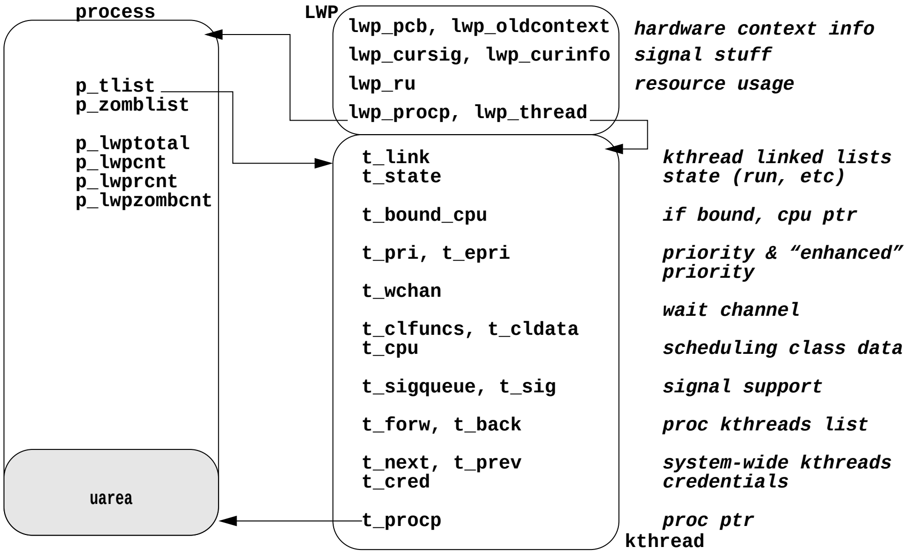
Kernel Process Model

- **Open file list in uarea**
  - Array of uf_entry structures, each structure contains a pointer to the file struct, a file flag field, and a reference count
  - SunOS 5.7 adds a poll cache structure pointer



Kernel Process Model

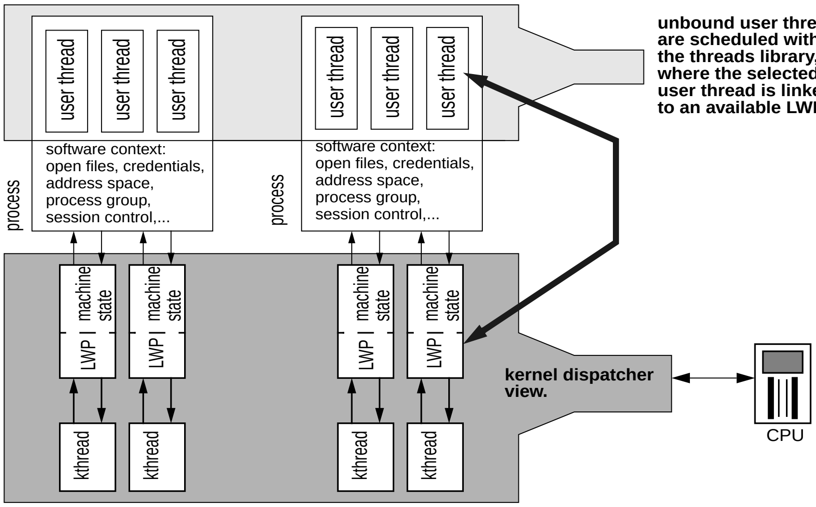
- **LWPs & kthreads**



Kernel Process Model

- **Kthreads/LWPs get scheduled independently of other kthread/LWPs in the same process**
- **User threads are scheduled by a threads library dispatcher**
  - A user thread gets scheduled by being placed on an LWP/Kthread
  - User threads have their own priority scheme
- **Kthread/LWP each have their own scheduling class data and priority**

Dispatcher Views



unbound user threads are scheduled within the threads library, where the selected user thread is linked to an available LWP.

kernel dispatcher view.

Kernel Process Model

- **Interesting departures from traditional UNIX...**
  - It’s the kernel thread (and it associated LWP) that gets put on a dispatch queue, given a priority and scheduling class, etc, not the process
  - Kernel threads and LWP’s within a process are visible with ps(1)

```
fawlty> ps -eLc
   PID   LWP  CLS PRI TTY     LTIME CMD
     0     1  SYS  96 ?        0:01 sched
     1     1   TS  58 ?        0:00 init
     2     1  SYS  98 ?        0:00 pageout
     3     1  SYS  60 ?       13:04 fsflush
   233     1   TS  58 ?        0:00 sendmail
   119     1   TS  58 ?        0:00 in.route
   317     1   TS  59 ?        0:00 sac
   182     1   TS  33 ?        0:00 syslogd
   182     2   TS  58 ?        0:00 syslogd
   182     3   TS  58 ?        0:00 syslogd
   182     4   TS  58 ?        0:00 syslogd
   182     5   TS  58 ?        0:00 syslogd
   182     6   TS  58 ?        0:00 syslogd
   156     1   TS  48 ?        0:00 inetd
```

Kernel Process Model

- **Process creation - the traditional fork/exec model is implemented for process creation**

```
main()
{
        pid_t pid;
        pid = fork();
        if (pid == 0) /* new child process */
            exec()
        else if ( pid > 0) /* parent */
            wait()
        else
            fork failed
}
```

Kernel Process Model

- **Process creation - a couple different “forks” available**
  - fork(2) - traditional behavior, replicates entire process, including all threads
  - fork1(2) - replicate the process and only the calling thread
  - vfork(2) - don’t replicate the address space - borrow it from the parent and get pages on exec
  - All thread ultimately enter kernel cfork() function

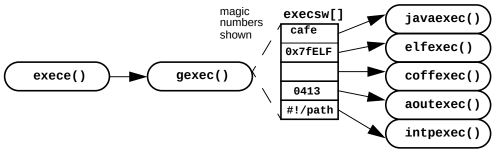
Kernel Process Model

```
cfork()
       kmem_alloc proc structure
       state to SIDL
       pid_assign()
          get a pid structure
          get /proc directory slot
          init pid struct
       check for proc table overflow (v.v_procs)
       check per-user limit
       put newproc on system-wide linked list
       set parent-child-sibling proc pointers
       copy profile state to child
       increment reference count on open files
       copy parent uarea to child
       if (vfork)
          set child address space from parent
       else
          as_dup()
       if (fork1())
```

```
       forklwp()
          lwp_create()
             thread_create()
else /* not fork1 */
       loop through p_tlist
          for each
             forklwp()
                lwp_create()
                   thread_create()
replicate scheduling call info from parent
add child to parent process group
set child process state to SRUN
if (vfork())
       cpu_sysinfo.vfork++
       continuwlwps()
else
       cpu_sysinfo.fork++
       put child ahead of parent on dispatch queue
return PID of child to parent
return 0 to child
```

Kernel Process Model

- **Time to exec**
  - exec(2) overlays new process with new program
  - SunOS supports several different executable file types
  - Object file specific vectoring to correct exec routine via switch table mechanism



Kernel Threads

- **Several kernel threads get created during the initialization process**
- **Most are daemons - placed on the system-wide linked list of kernel threads**
- **They’re all SYS class threads**
- **They’re unique in that they do not have an associated LWP, or process**
- **The kthread structure itself contains most of the necessary context state - the kernel stack & hardware context**

Kernel Threads

thread_reaper() - a daemon. Cleanup zombie threads on deathrow.

mod_uninstall_daemon() - module unloads for CPR

hotplug_daemon() - Device hotplug support

kmem_async_thread() - slab allocator garbage collector

seg_pasync_thread() - pagelock pages reclaim

ksyms_update_thread() - Keep /dev/ksyms current

Kernel Threads

callout_thread() - callout queue processing

cpu_pause() - per processor. Put the processor in a safe place for offline.

modload_thread() - kernel module load

hwc_parse_thread() - read driver.conf file

- STREAMS

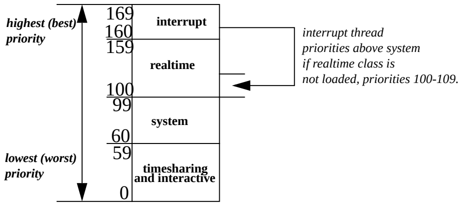
background() - Service STREAM queues

freebs() - Manage free list of message blocks

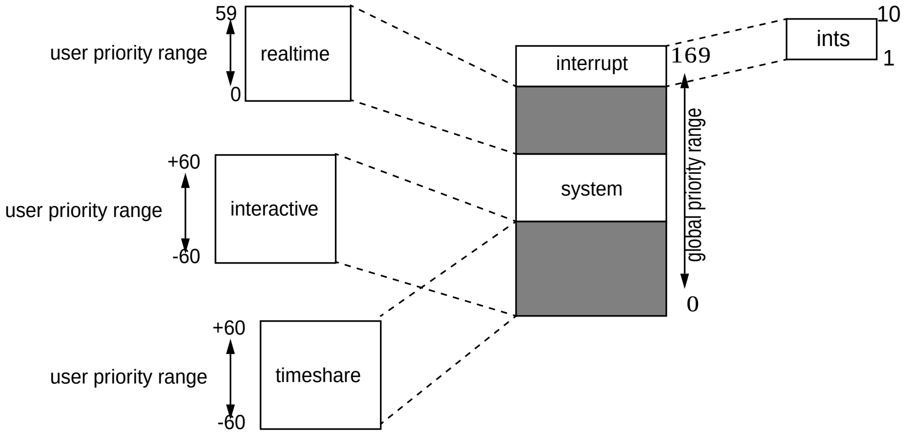
qwriter_outer_thread() - Process out syncq messages

Scheduling Classes

- **SunOS implements scheduling classes, where a specific class defines the priority range and policies applied to the scheduling of kernel threads on processors**
- **Timeshare (TS), Interactive (IA), System (SYS) and Realtime (RT) classes defined**


highest (best) priority
lowest (worst) priority
169
160
159
100
99
60
59
0
interrupt
realtime
system
timesharing and interactive
interrupt thread priorities above system if realtime class is not loaded, priorities 100-109.

Scheduling Classes



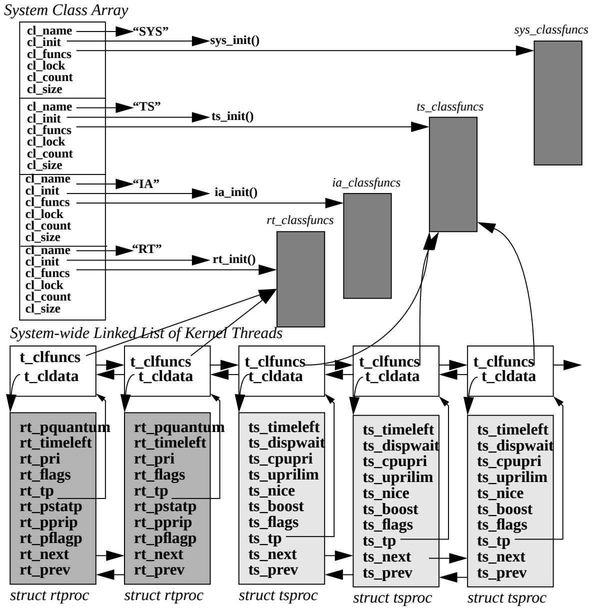
Quick Tidbit

- **Use dispadmin(1M) or /etc/crash for scheduling class info**

```
# dispadmin -l
CONFIGURED CLASSES
==================

SYS     (System Class)
TS      (Time Sharing)
IA      (Interactive)
# /etc/crash
dumpfile = /dev/mem, namelist = /dev/ksyms, outfile = stdout
> class
SLOT    CLASS   INIT FUNCTION   CLASS FUNCTION

0       SYS     100cdaec        1042835c
1       TS      100fd13c        104390dc
2       IA      100fd214        1043913c
>
```

- Note the RT class is not loaded

Scheduling Classes



Scheduling Classes

- **Each class has a class-specific data structure and functions table**
- **Dispatch tables are implemented that provide the values used to calculate and re-adjust thread priorities**
- **TS & IA threads share the same dispatch table**
- **There’s a RT thread dispatch table**
- **SYS threads do not need a dispatch table, since the rules do not apply**

Scheduling Class Specific Functions

- **Implemented via macros**

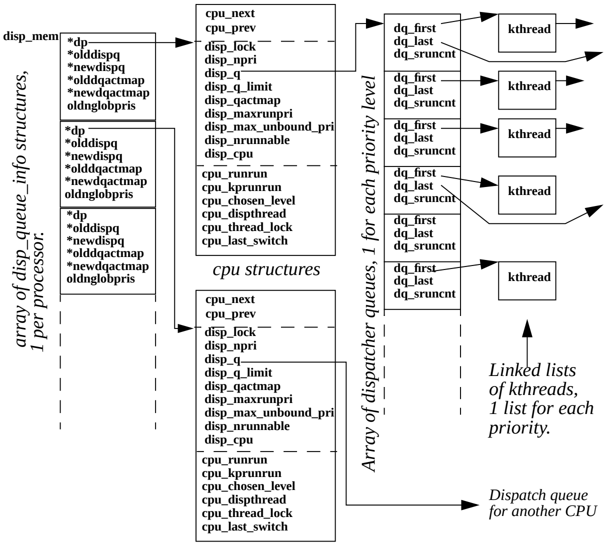
```
#define CL_ENTERCLASS(t, cid, clparmsp, credp, bufp) \
        (sclass[cid].cl_funcs->thread.cl_enterclass) (t, cid, \
            (void *)clparmsp, credp, bufp)
```

- **Class management and priority manipulation functions**
  - xx_preempt, xx_sleep, xx_tick, xx_trapret, xx_fork, xx_parms[get|set], xx_donice, etc

Dispatch Queues

- **SunOS implements per-processor dispatch queues - actually a queue of queues**
- **Several dispatcher-related variables maintained in the CPU structure as well**
  - cpu_runrun - cpu_kprunrun - preemption flags
  - cpu_disp - dispatcher data and root of queues
  - cpu_chosen_level - priority of next selected thread
  - cpu_dispthread - kthread pointer

Dispatch Queues



Thread Priorities & Scheduling

- **Priority inherited from parent, alterable via priocntl(1) command or system call**
- **Typically, threads run as either TS or IA threads**
  - IA threads created when thread is associated with a windowing system
- **RT threads are explicitly created**
- **SYS class used by kernel threads, and for TS/IA threads when a higher priority is warranted**
- **Interrupts run at interrupt priority**

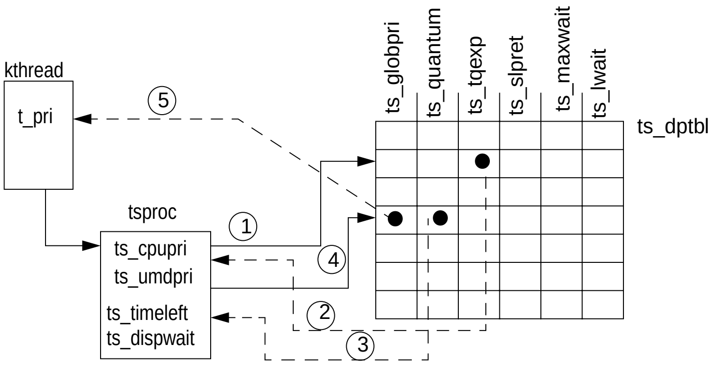
Thread Priorities and Scheduling

- **Kernel implements preemption mechanism for RT support**
  - There’s 2 preemption levels - user preemption and kernel preemption
  - Seperate dispatch queue for threads at a priority above kernel preemption (RT threads can preempt the kernel)

TS Dispatch Table

```
# 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       #      1
       160          0        51           0        51       #     10
       120         10        52           0        52       #     20
        80         20        53           0        53       #     30
        40         30        55           0        55       #     40
        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
```

Setting A TS/IA Priority

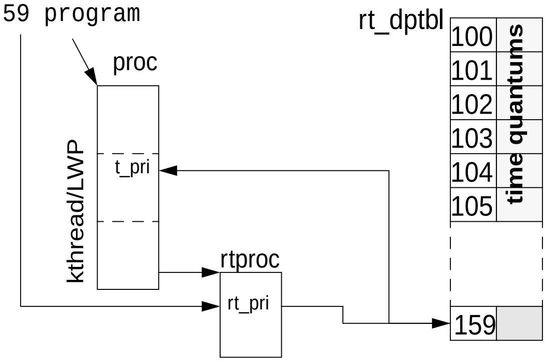


RT Dispatch Table

```
# Real Time Dispatcher Configuration
RES=1000

# TIME QUANTUM                    PRIORITY
# (rt_quantum)                      LEVEL
      1000                    #        0
       800                    #       10
       600                    #       20
       400                    #       30
       200                    #       40
       100                    #       50
```

Setting A RT Thread’s Priority

```
#priocntl -e -c RT -p 59 program
```



Thread Queue Placement

```
if (thread is bound to CPU-n) && (pri < kpreemptpri)
        CPU-n dispatch queue
if (thread is bound to CPU-n) && (pri >= kpreemptpri)
        CPU-n dispatch queue
if (thread is not bound) && (pri < kpreemptpri)
        place thread on a CPU dispatch queue
if (thread is not bound) && (pri >= kpreemptpri)
        place thread on cp_kp_queue
```

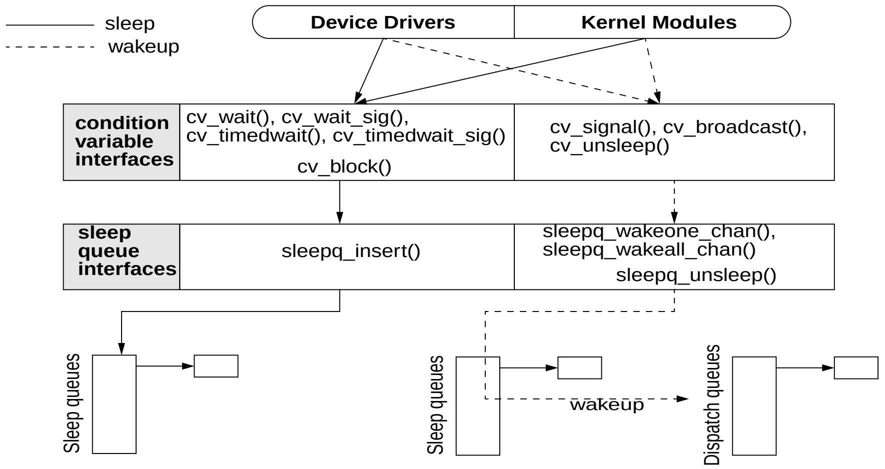
Dispatcher Functions

- **Queue manipulation**
  - setfrontdq(), setbackdq()
- **kernel thread selection**
  - swtch()
  - Code implements select & ratify

Sleep/Wakeup

- **The kernel supports sets of sleep queues for sleeping threads**
- **Condition variables are the synchronization object used to manage sleep/wakeup**
- **A condition variable represents an event or resource that a thread is waiting (sleeping) for**
- **The address of the condition variable is stored in the wchan field of the kernel thread**
- **Remember - turnstiles are used for sleeps on mutexes & R/W locks**

Sleep/Wakeup Kernel Subsystem



sleep

wakeup

Device Drivers | **Kernel Modules**

condition variable interfaces	cv_wait(), cv_wait_sig(), cv_timedwait(), cv_timedwait_sig()
	cv_block()
		cv_signal(), cv_broadcast(), cv_unsleep()

sleep queue interfaces	sleepq_insert()	sleepq_wakeone_chan(), sleepq_wakeall_chan()
		sleepq_unsleep()

Sleep queues

Sleep queues

wakeup

Dispatch queues

Signals

- **A signal is a means of interrupting or notifying a process or thread of an event**
- **Signals have been implemented in UNIX for about as long as we’ve had UNIX**
- **Original implementation was unreliable**
  - signal disposition was reset to default upon entering handler
  - reliable signals appeared in SVR4
  - “unreliable” signals still possible in SunOS, depending on which API is used

Signals

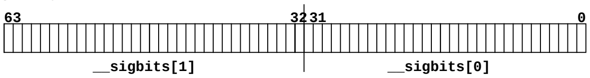
- **The kernel defines 42 signals as of SunOS 5.7**
- **Every signal has a unique name, SIGxxx, and a unique signal number**
- **There are several possible actions that a process or thread can take as a result of receiving a signal**
  - Ignore
  - Catch
  - Terminate
  - Terminate with extreme prejudice (core dump)

Signals

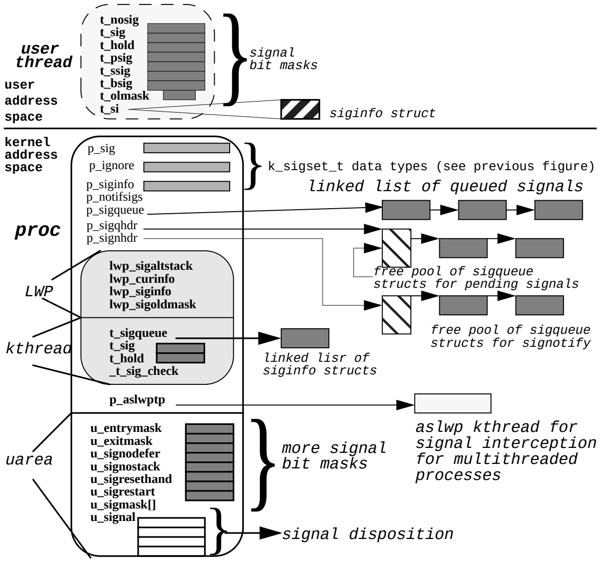
- **Signals represented as bits in a data structure**
- **For each signal, there is a corresponding bit in a signal mask**

```
typedef struct {
      unsigned long __sigbits[2];
} k_sigset_t;
```

signals posted by setting bit number that corresponds to signal number



Signals

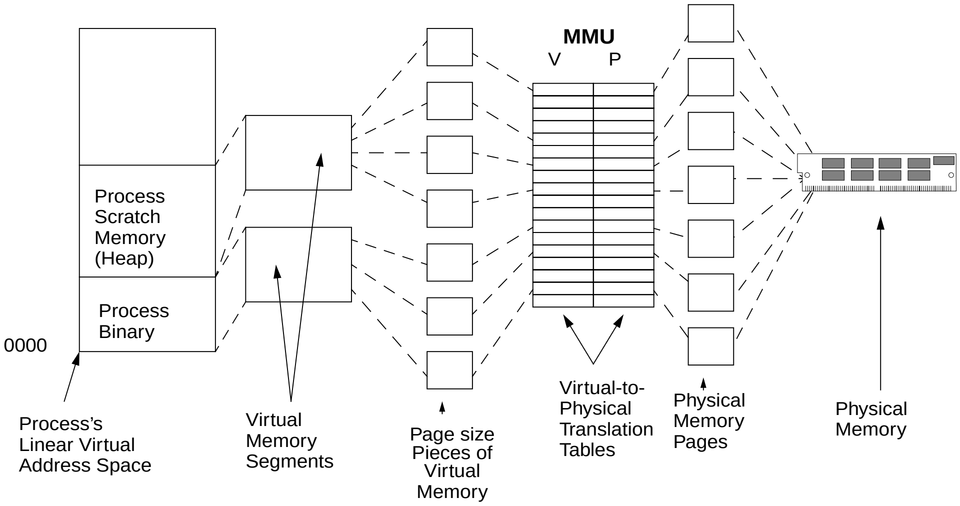


Interprocess Communication (IPC)

- **Traditional System V facilities**
  - Shared Memory, Message Queues, Semaphores
- **Provide process-to-process communication path and synchronization**
- **Facilities extended as part of POSIX**
  - Shared Memory, Message Queues, Semaphores
  - Sys V & POSIX are the same, only different

Virtual Memory

The Solaris Memory Model



MMU

V P

Process Scratch Memory (Heap)

Process Binary

0000

Process's Linear Virtual Address Space

Virtual Memory Segments

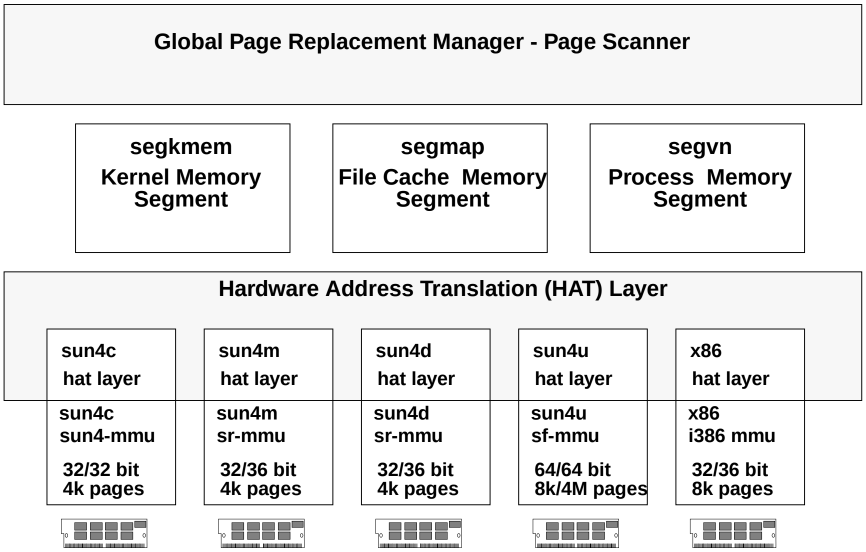
Page size Pieces of Virtual Memory

Virtual-to-Physical Translation Tables

Physical Memory Pages

Physical Memory

Solaris Memory Architecture



Global Page Replacement Manager - Page Scanner

segkmem
Kernel Memory Segment

segmap
File Cache Memory Segment

segvn
Process Memory Segment

Hardware Address Translation (HAT) Layer

sun4c hat layer — sun4c sun4-mmu — 32/32 bit 4k pages

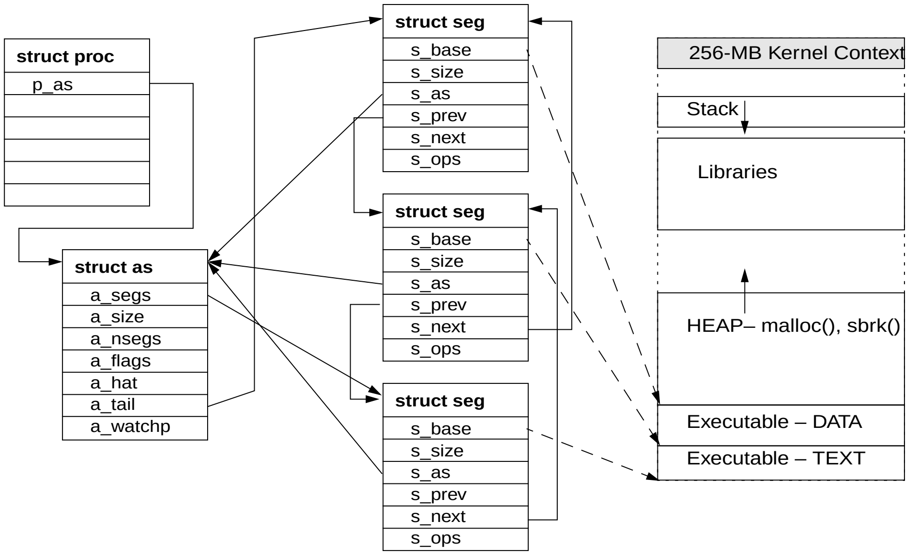
sun4m hat layer — sun4m sr-mmu — 32/36 bit 4k pages

sun4d hat layer — sun4d sr-mmu — 32/36 bit 4k pages

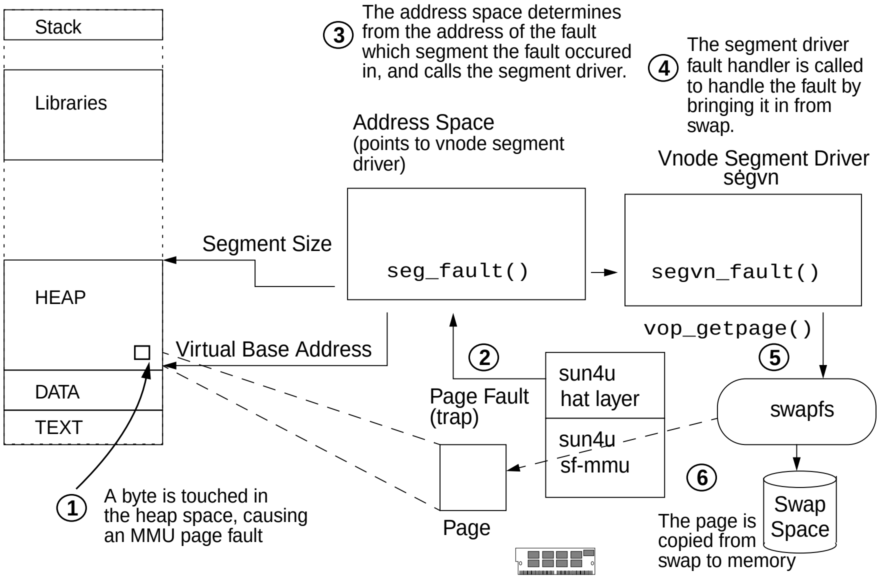
sun4u hat layer — sun4u sf-mmu — 64/64 bit 8k/4M pages

x86 hat layer — x86 i386 mmu — 32/36 bit 8k pages

Segments and Addr. Spaces



An example of a memory segment



1 A byte is touched in the heap space, causing an MMU page fault

3 The address space determines from the address of the fault which segment the fault occured in, and calls the segment driver.

4 The segment driver fault handler is called to handle the fault by bringing it in from swap.

6 The page is copied from swap to memory

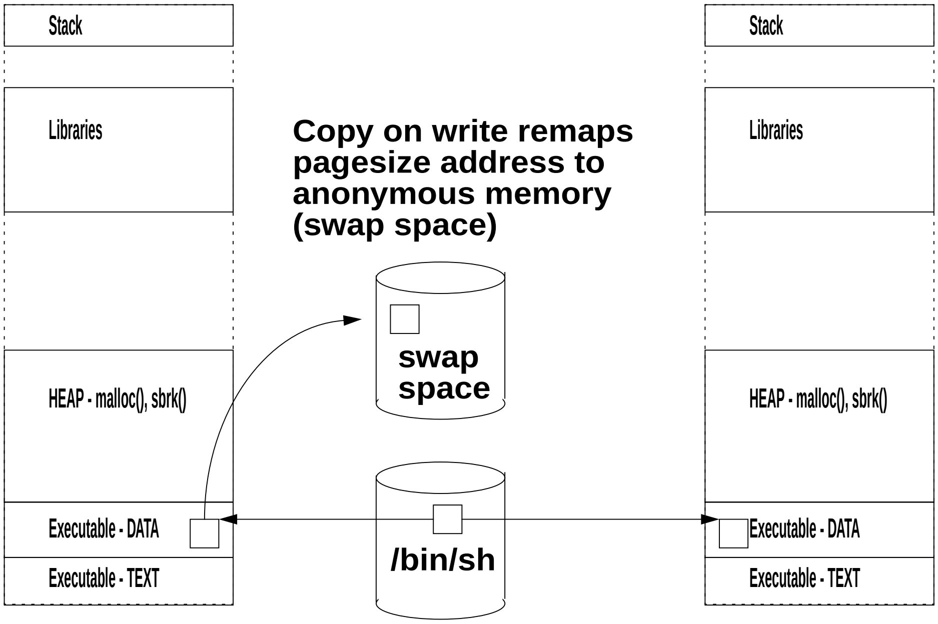
Page allocation

- **Pages are allocated into address space on demand**
  - Anonymous memory (heap) virtual address space is empty until first referenced
  - A page fault is generated the first time memory is accessed
  - The page fault realizes this is the first reference and allocated a zeroed page at that address
  - This is known as zero-fill-on-demand (ZFOD)

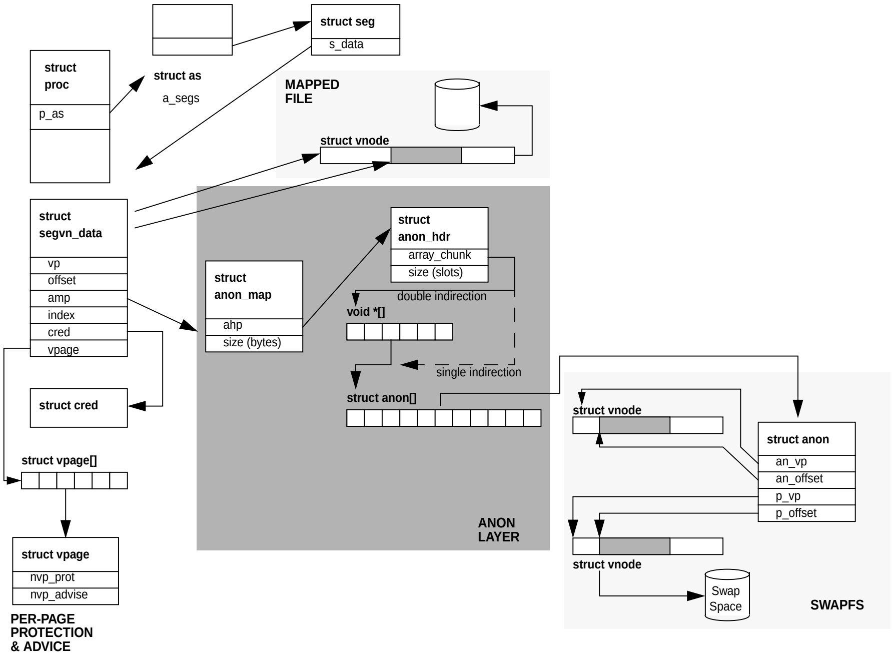
Page Sharing

- **Pages may be shared between segments**
  - e.g. multiple processes may map /bin/sh
  - Each segment has its own TLB mappings
- **Pages may be shared private/public**
  - Public sharing makes modified pages visible to all
  - Private sharing makes modified pages local
  - Private sharing is done via copy-on-write (COW)

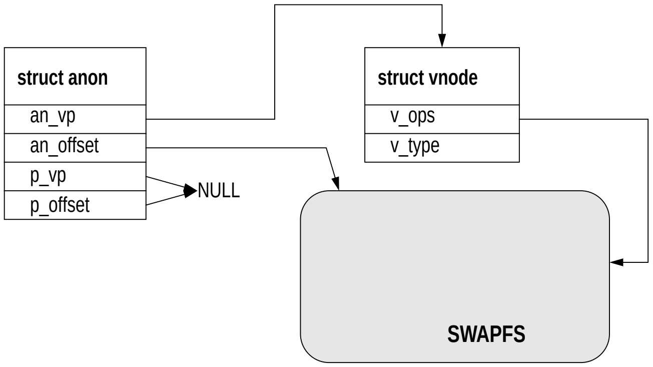
The Copy On Write (COW)



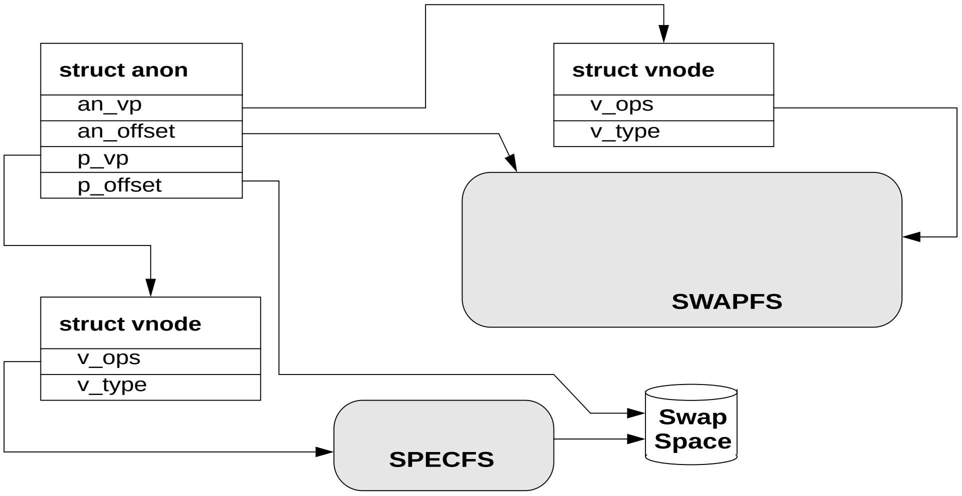
Copy on write remaps pagesize address to anonymous memory (swap space)



SWAPFS



SWAPFS

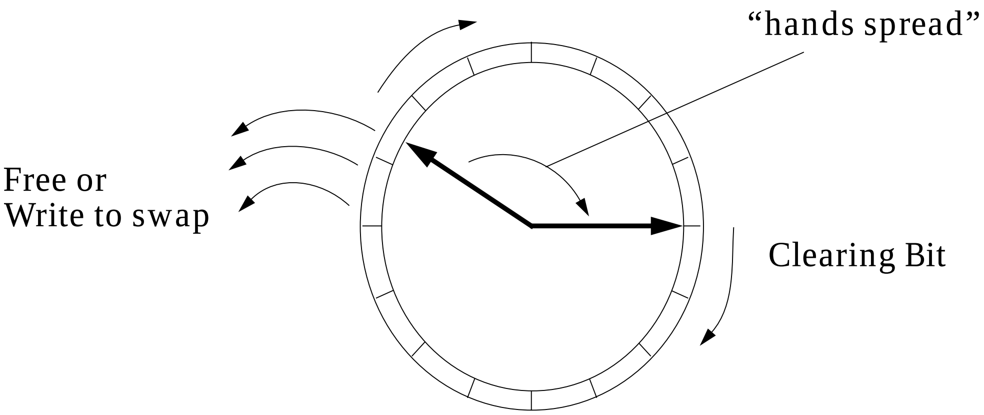


Global Memory Management

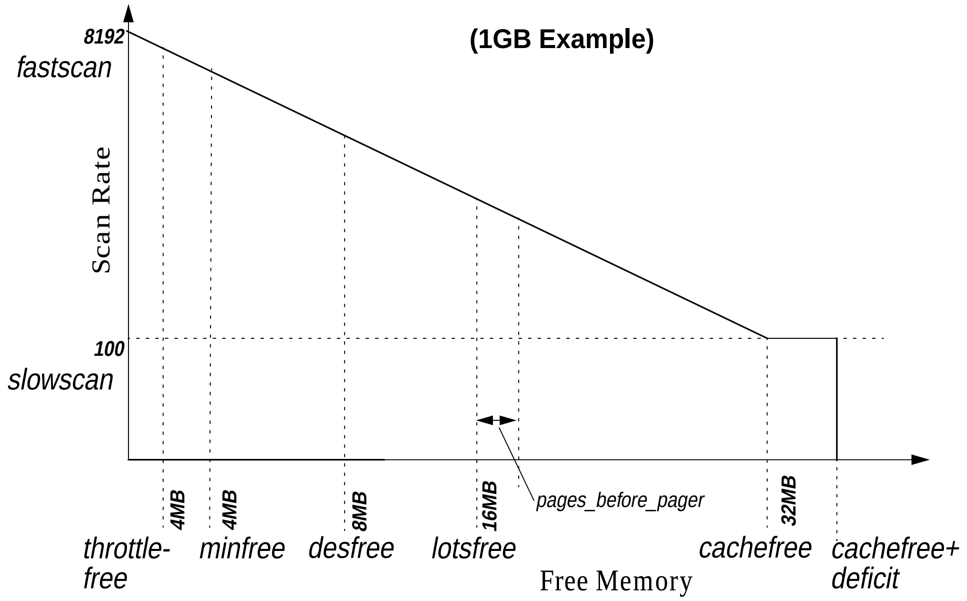
- **Demand Paged**
  - Not recently used (NRU) algorithm
- **Dynamic file system cache**
  - Where has all my memory gone?
- **Page scanner**
  - Operates bottom up from physical pages
  - Default mode treats all memory equally

Global Memory Management

- **Demand Paging**
  - Not Recently Used (LRU) Algorithm



Global Paging Dynamics



Priority Paging

- **Solaris 7 FCS or Solaris 2.6 with T-105181-09**
  - http://www.sun.com/sun-on-net/performance/priority_paging.html
  - Set priority_paging=1 or cachefree in /etc/system
- **Solaris 7 Extended vmstat**
  - ftp://playground.sun.com/pub/rmc/memstat
- **Solaris 8**
  - New VM system, priority paging implemented at the core (make sure it’s disabled in Sol 8!)
  - New vmstat flag, “-p”

LRU Algorithm

- **Use vmstat or the memstat command on Solaris 7**
  - ftp://playground.sun.com/pub/rmc/memstat

```
# vmstat 3

 procs     memory           page           disk         faults      cpu
 r b w   swap  free  re  mf pi po fr de sr f0 s0 s4 s6   in   sy   cs us sy id
 0 0 0 269776  21160   0   0  0  0  0  0  0  0  0  0  2  154  200   92  0  0 100
 0 0 0 269776  21152   0   0  0  0  0  0  0  0  0  0  2  155  203  113  0  0 99
 0 0 0 269720  3896   5  17 80  0 109 0 59  0  0  0  2  221  773  134  0  2 98
 0 0 0 269616  3792   0    0 160 0 160 0 76  0  0  0  2  279  242  130  0  1 99
 0 0 0 269616  3792   0    0 192 0 192 0 105 0  0  0  2  294  225  138  0  1 99
 0 0 0 269616  3800   1  90 234 5 232 0 99  0  0  0  2  323  964  305  5  3 92
 0 0 0 269656  3832   0    0 106 0 106 0 51  0  0  0  2  237  212  121  0  1 99
```

```
# memstat 3

memory  ---------- paging ----------- - executable -  - anonymous -  -- filesys -- --- cpu ---
  free  re  mf   pi   po   fr  de  sr  epi  epo  epf  api  apo  apf  fpi  fpo  fpf us sy wt id
 21160   0  22    0    5    5   0   0    0    0    0    0    0    0    0    5    5  0  1  0 99
 21152   0   0    0    0    0   0   0    0    0    0    0    0    0    0    0    0  0  0  0 100
 21152   0  18   34    2    2   0   0    0    0    0    0    0    0   34    2    2  0  1  0 99
 11920   0   0  277  106  272   0 153    0    0   32    0   98  149  277    8   90  0  3  0 97
 11888   0   0  256   69  224   0 106    0    0   16    0   69  178  256    0   29  0  3  1 96
 11896   0   0  213  106  261   0 124    0    0   26    0  106  232  213    0    2  0  3 13 84
 11904   0   0  245   66  242   0 122    0    0   16    0   64  221  245    2    5  0  2  0 98
 11896   0   0  245   64  224   0 132    0    0   21    0   64  189  245    0   13  0  2  0 98
```

Simple Memory Rule:

- **Identifying a memory shortage without PP:**
  - Scanner not scanning -> no memory shortage
  - Scanner running, page ins and page outs, swap device activity -> potential memory shortage
  - (use separate swap disk or 2.6 iostat -p to measure swap partition activity)
- **Identifying a memory shortage with PP on Sol 7:**
  - api and apo should be zero in memstat, non zero is a clear sign of memory shortage
- **Identifying a memory shortage on Sol 8:**
  - scan rate != 0

Intimate Shared Memory

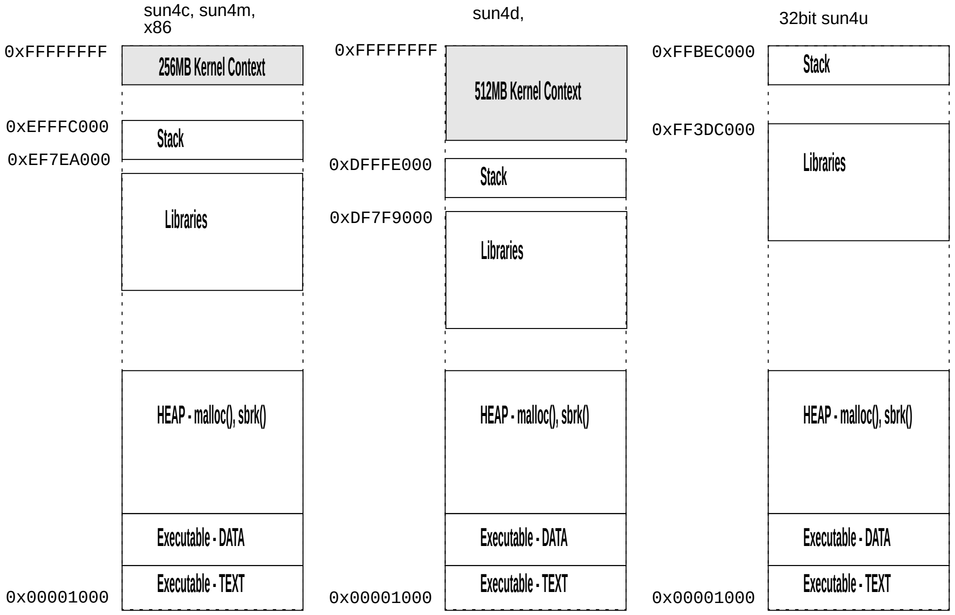
- **The Virtual to Physical page translation tables are only valid for one address space**
  - Each time we context switch to another process, we need to reload the TLB/TSB
  - For databases that share 90% of their address space between processes, this is a large overhead
- **Sharing Page Tables**
  - A special type of shared memory in Solaris is used for databases
  - Intimate Shared Memory - ISM.
  - Invoke with an additional flag to shmat () - SHARE_MMU
  - ISM also uses large 4M pages on Solaris 2.6 ->4M pages may become fragmented, shared memory must be allocated at boot time before the freelist becomes empty

Memory Analysis

- **The ps command**

```
# ps -ale

USER       PID %CPU %MEM   SZ  RSS TT       S    START   TIME COMMAND
root     22998 12.0  0.8 4584 1992 ?        S 10:05:30   3:22 /usr/sbin/nsr/nsrc
root     23672  1.0  0.7 1736 1592 pts/16   O 10:22:54   0:00 /usr/ucb/ps -aux
root         3  0.4  0.0    0    0 ?        S   Sep 28 166:38 fsflush
root       733  0.4  1.0 6352 2496 ?        S   Sep 28 174:29 /opt/SUNWsymon/jre
root       345  0.3  0.7 2968 1736 ?        S   Sep 28  55:39 /usr/sbin/nsr/nsrd
root     23100  0.2  0.5 3880 1104 ?        S   Oct 15   0:25 rpc.rstatd
root       732  0.2  2.5 9920 6304 ?        S   Sep 28  94:43 esd - init topolog
```

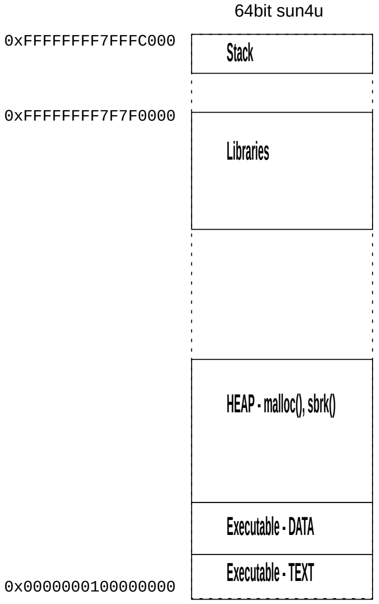
32 bit Address Space Layout



32 bit limits

- **Solaris 2.5**
  - Heap is limited to 2GB, malloc will fail beyond 2GB
- **Solaris 2.5.1**
  - Heap limited to 2GB by default
  - Can go beyond 2GB with kernel patch 103640-08+
  - can raise limit to 3.75G by using ulimit or rlimit() if uid=root
  - Do not need to be root with 103640-23+
- **Solaris 2.6**
  - Heap limited to 2GB by default
  - can raise limit to 3.75G by using ulimit or rlimit()
- **Solaris 7 & 8**
  - Limits are raised by default
  - 32 bit program can malloc 3.99GB

64 bit Address Space Layout



64bit sun4u

0xFFFFFFFF7FFFC000 Stack

0xFFFFFFFF7F7F0000 Libraries

HEAP - malloc(), sbrk()

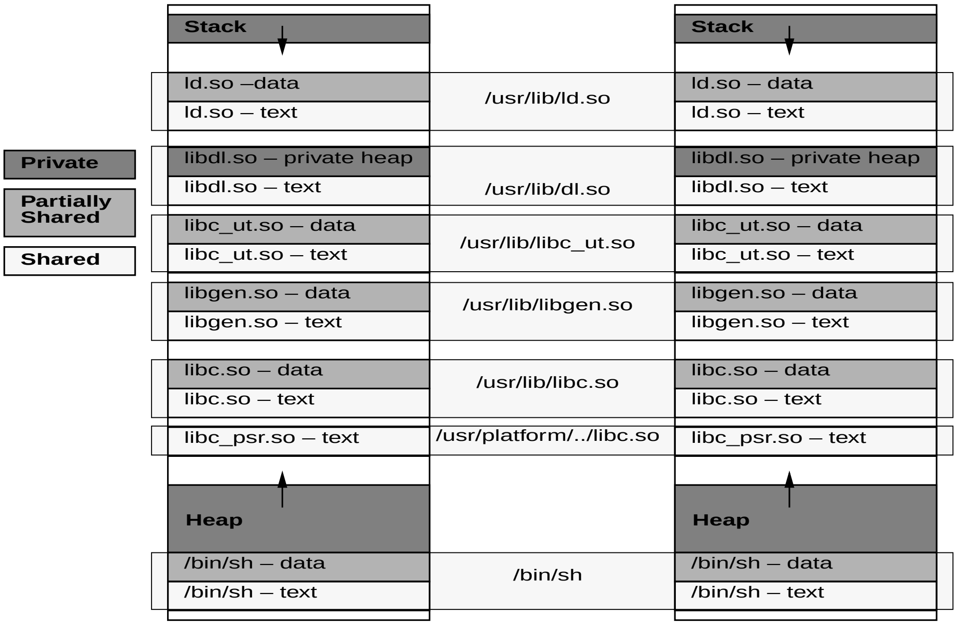
Executable - DATA

Executable - TEXT

0x0000000100000000

- **No 3.99GB limits!**
  - Processes can malloc() beyond 3.99GB when compiled in 64 bit mode
- **$ cc -xarch=v9**

•



Private

Partially Shared

Shared

Stack | | Stack

ld.so –data | /usr/lib/ld.so | ld.so – data

ld.so – text | | ld.so – text

libdl.so – private heap | /usr/lib/dl.so | libdl.so – private heap

libdl.so – text | | libdl.so – text

libc_ut.so – data | /usr/lib/libc_ut.so | libc_ut.so – data

libc_ut.so – text | | libc_ut.so – text

libgen.so – data | /usr/lib/libgen.so | libgen.so – data

libgen.so – text | | libgen.so – text

libc.so – data | /usr/lib/libc.so | libc.so – data

libc.so – text | | libc.so – text

libc_psr.so – text | /usr/platform/../libc.so | libc_psr.so – text

Heap | | Heap

/bin/sh – data | /bin/sh | /bin/sh – data

/bin/sh – text | | /bin/sh – text

The pmap command

```
# pmap -x 23532

23532:  /bin/sh
Address   Kbytes Resident Shared Private Permissions       Mapped File
00010000      88       88     88       - read/exec         sh
00034000       8        8      8       - read/write/exec   sh
00036000      16       16      -      16 read/write/exec   [ heap ]
EF6C0000      16       16     16       - read/exec         en_US.so.1
EF6D2000       8        8      8       - read/write/exec   en_US.so.1
EF6E0000      16       16     16       - read/exec         libc_psr.so.1
EF700000     592      520    504      16 read/exec         libc.so.1
EF7A2000      24       24      8      16 read/write/exec   libc.so.1
EF7A8000       8        -      -       - read/write/exec   [ anon ]
EF7B0000       8        8      8       - read/exec/shared  libdl.so.1
EF7C0000     112      112    112       - read/exec         ld.so.1
EF7EA000      16       16      8       8 read/write/exec   ld.so.1
EFFFC000      16       16      -      16 read/write/exec   [ stack ]
--------  ------  ------  ------  ------
total Kb     928      848    776      72
```

MemTool

- **What MemTool is and how to get it**
- **System Memory Summary**
- **File system page cache**
- **Process Memory usage**

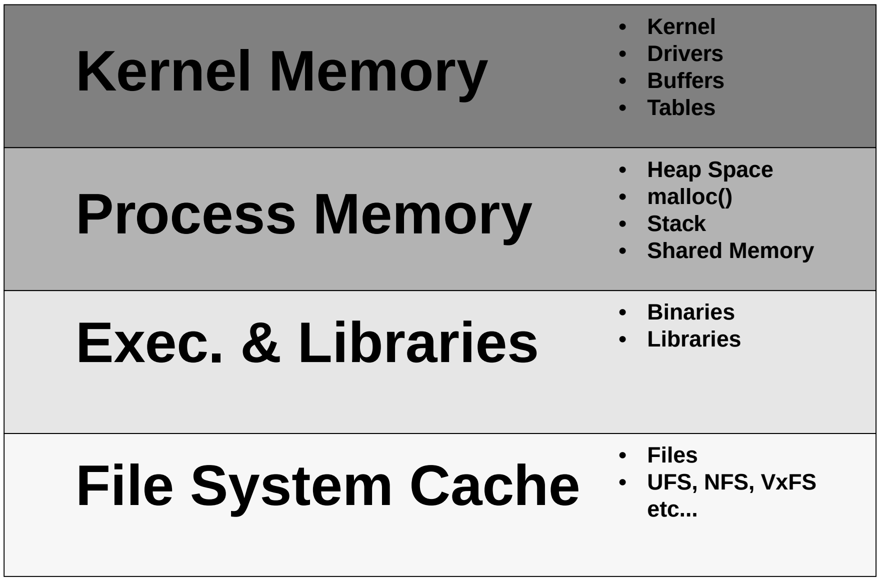
Memtool

- **Prototype developed to allow memory sizing and capacity planning for Solaris**
- **Loadable kernel memory module**
- **Tested, but unsupported by Sun**
- **GUI & CUI**
  - memtool, mem
- **Commands**
  - prtmem, pmem, memps

Where to get it

- **memtool-request@chessie.eng.sun.com**
- **SPARC**
  - Solaris 2.6, 7, 8
- **Intel**
  - Solaris 2.6, 7, 8

Different memory categories:



Kernel Memory

- Kernel
- Drivers
- Buffers
- Tables

Process Memory

- Heap Space
- malloc()
- Stack
- Shared Memory

Exec. & Libraries

- Binaries
- Libraries

File System Cache

- Files
- UFS, NFS, VxFS etc...

System Memory Summary

```
# prtmem

Total memory:            3879 Megabytes
Kernel Memory:            120 Megabytes
Application:             3263 Megabytes
Executable & libs:         23 Megabytes
File Cache:                18 Megabytes
Free, file cache:          43 Megabytes
Free, free:               410 Megabytes

#

# prtmem

Total memory:             492 Megabytes
Kernel Memory:             25 Megabytes
Application:              120 Megabytes
Executable & libs:         44 Megabytes
File Cache:                 9 Megabytes
Free, file cache:          56 Megabytes
Free, free:               237 Megabytes

#
```

Note that the *prtmem* command is only available with the unbundled MemTool package - to obtain, email memtool-request@chessie.eng.sun.com

Filesystem page cache

```
# memps -m

SunOS devhome 5.7 SunOS_Development sun4u    05/03/99

00:34:37
   Size E/F Filename
 16040k F   /ws/on28-gate/usr/src/uts/cscope.out       4GB E4000 Server
  8384k E   /export/ws/dist/share/netscape,v4.06/5bin.sun4/netscape
  5776k E   /export/ws/dist/share/framemaker,v5.5.3/bin/sunxm.s5.sparc/maker5X.e
  4440k E   /ws/on297-tools/SUNWspro/SC5.x/contrib/XEmacs20.3-b91/bin/sparc-sun-
  4160k E   /export/ws/dist/share/bugtraq_plus,v1.0.8/5bin.sun4/_progres
  3856k F   /var/crash/grafspee/vmcore.0
  2408k E   /ws/on297-tools/SUNWspro/SC5.x/WS5.0/bin/workshop
  2040k E   /export/ws/dist/share/acroread,v3.01/Reader/sparcsolaris/lib/libXm.s
  1712k E   /usr/dt/lib/libXm.so.4
  1464k E   /usr/dt/lib/libXm.so.3
  1312k E   /usr/openwin/server/lib/libserverdps.so.5
  1072k E   /usr/lib/sgml/nsgmls
   968k E   /ws/on297-tools/SUNWspro/SC5.x/SC5.0/bin/acomp
   896k E   /export/ws/dist/share/acroread,v3.01/Reader/sparcsolaris/lib/libread
   840k E   /export/ws/dist/share/acroread,v3.01/Reader/sparcsolaris/bin/acrorea
   776k E   /ws/on297-tools/SUNWspro/SC5.x/WS5.0/lib/eserve
   736k E   /usr/lib/sparcv9/libc.so.1
   680k E   /usr/lib/libc.so.1
   648k E   /opt/SUNWvmsa/jre/lib/sparc/green_threads/libjava.so
   616k E   /export/ws/local/bin/irc
   608k E   /usr/openwin/bin/Xsun
   584k F   /export/ws/dist/share/bugtraq_plus,v1.0.8/patch/patch_001/common/bug
   512k E   /1d80068:  183021
   504k E   /usr/lib/libnsl.so.1
   496k E   /usr/dt/bin/dtwm
```

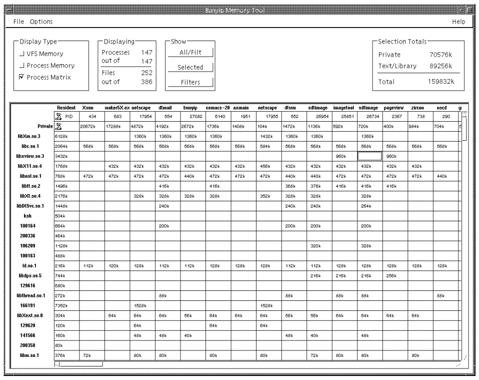
512MB U60 desktop

Process Memory - memps

```
# memps

SunOS chessie 5.6 Generic sun4u    10/16/98

10:24:56
   PID      Size Resident   Shared  Private  Process
    732    9920k    7160k    2216k    4944k  esd - init topology -dir /var/opt/S
    731   10432k    6168k    2280k    3888k  esd - init event -dir /var/opt/SUNW
    729    7752k    5744k    2184k    3560k  esd - init trap -dir /var/opt/SUNWs
    730    7344k    5624k    2112k    3512k  esd - init cfgserver -dir /var/opt/
  21071    5808k    3800k    1184k    2616k  nwadmin
    733    6352k    3744k    1304k    2440k  /opt/SUNWsymon/jre1.1.6/bin/../bin/
    396    5840k    2752k    1368k    1384k  /usr/lib/nfs/mountd
  22998    4584k    2344k    1120k    1224k  /usr/sbin/nsr/nsrck -M chessie
   3447    4472k    2640k    1616k    1024k  imapd
    345    2968k    2384k    1408k     976k  /usr/sbin/nsr/nsrd
  17049    3568k    2216k    1280k     936k  /usr/sbin/nscd
    295    2192k    2040k    1144k     896k  /usr/lib/sendmail -bd -q1h

(ctd...)
```

The MemTool GUI

- **File system page cache**
- **Process summary and detail**
- **Process Matrix**



Display Type

□ VFS Memory
□ Process Memory
☑ Process Matrix

Displaying

Processes 147
out of 147
Files 252
out of 386

Selection Totals

Private 70576k
Text/Library 89256k
Total 159832k

	Resident	Xsun	maker5X.ex	netscape	dtmail	bunyip	xemacs-20	axmain	netscape	dtwm	sdtimage	imagetool	sdtimage	pageview	zircon	nscd	g
	PID	434	683	17954	554	27082	6140	1951	17955	552	26954	25851	26734	2367	738	290	
Private		20672k	17288k	4872k	4192k	2672k	1736k	1408k	104k	1472k	1136k	592k	720k	400k	984k	704k	5
libXm.so.3	6128k			1360k	1360k	1360k	1360k		1432k	1360k	1360k		1360k				
libc.so.1	2064k	568k	568k	568k	568k	568k	568k	568k	584k	568k	568k	568k	568k	568k	568k	568k	
libxview.so.3	3432k											960k		960k			
libX11.so.4	1768k		432k	432k	432k	432k	432k	432k	456k	432k	432k	432k	432k	432k	432k		
libnsl.so.1	768k	472k	472k	472k	472k	440k	472k	472k	472k	440k	440k	472k	472k	472k	472k	440k	
libtt.so.2	1496k				416k		416k			368k	376k	416k	416k	416k			
libXt.so.4	2176k			328k	328k	328k	328k		352k	328k	328k		328k				
libDtSvc.so.1	1448k				240k					240k	240k		264k				
ksh	504k																
100164	664k				200k					200k	200k		200k				
200336	464k																
106209	1128k										320k		328k				
100163	488k																
ld.so.1	216k	112k	120k	128k	112k	112k	128k	128k	128k	112k	112k	128k	128k	128k	128k	128k	
libdps.so.5	744k										216k	216k	216k	256k			
129616	680k																
libthread.so.1	272k				88k					88k		88k	88k			88k	
166191	7352k			1528k					1528k								
libXext.so.0	304k		64k	64k	64k	56k	64k	64k	64k	56k	56k	64k	64k	64k	64k		
129620	120k			64k			64k		64k								
141566	160k			48k	48k	40k				48k	40k		48k				
200358	80k																
libm.so.1	376k	72k		80k	80k		80k		80k		72k	80k	80k		80k		

SWAP Space

- **Memory has two major swap states:**
  - Reserved - When memory is malloced but not referenced
  - Allocated - Once memory is accessed
  - (Unless MAP_NORESERVE)
- **You need enough swap for the amount of non-shared virtual memory space you use**

SWAP Space ctd...

```
# swap -s
total: 101456k bytes allocated + 12552k reserved = 114008k used, 597736k available
```

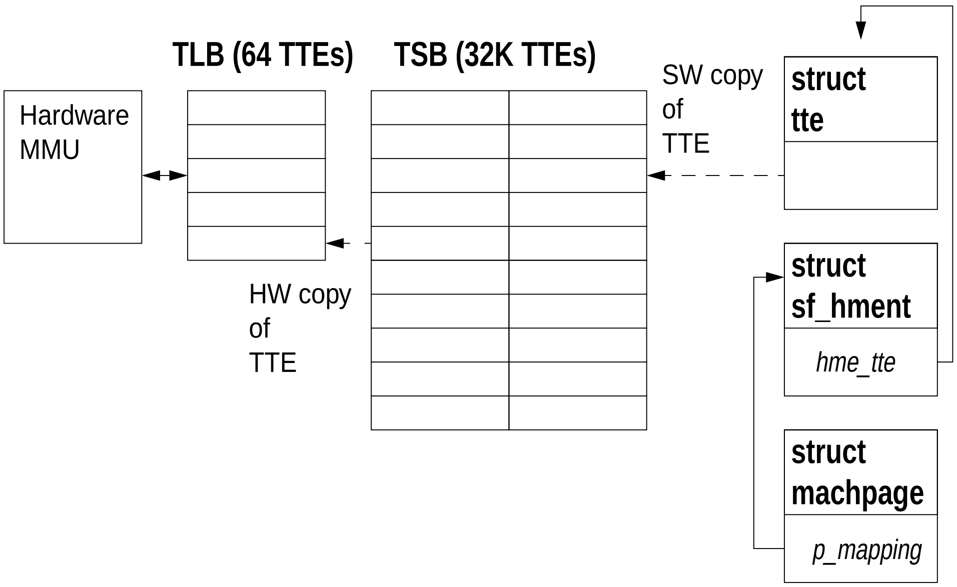
should read:

```
total: 101456k bytes unallocated + 12552k allocated = 114008k reserved, 597736k available
```

Swap:

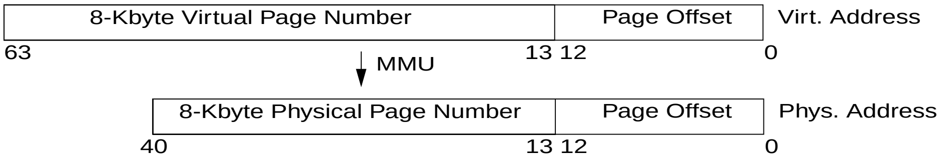
```
# ./prtswap -l
Swap Reservations:
--------------------------------------------------------------------------
Total Virtual Swap Configured:                              767MB =
RAM Swap Configured:                                            255MB
Physical Swap Configured:                               +       512MB

Total Virtual Swap Reserved Against:                        513MB =
RAM Swap Reserved Against:                                        1MB
Physical Swap Reserved Against:                         +       512MB

Total Virtual Swap Unresv. & Avail. for Reservation:        253MB =
Physical Swap Unresv. & Avail. for Reservations:                  0MB
RAM Swap Unresv. & Avail. for Reservations:             +       253MB

Swap Allocations: (Reserved and Phys pages allocated)
--------------------------------------------------------------------------
Total Virtual Swap Configured:                              767MB
Total Virtual Swap Allocated Against:                       467MB

Physical Swap Utilization: (pages swapped out)
--------------------------------------------------------------------------
Physical Swap Free (should not be zero!):                   232MB =
Physical Swap Configured:                                       512MB
Physical Swap Used (pages swapped out):                 -       279MB
```

Hardware Translation

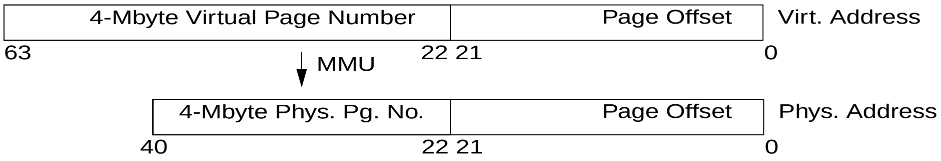


TTEs

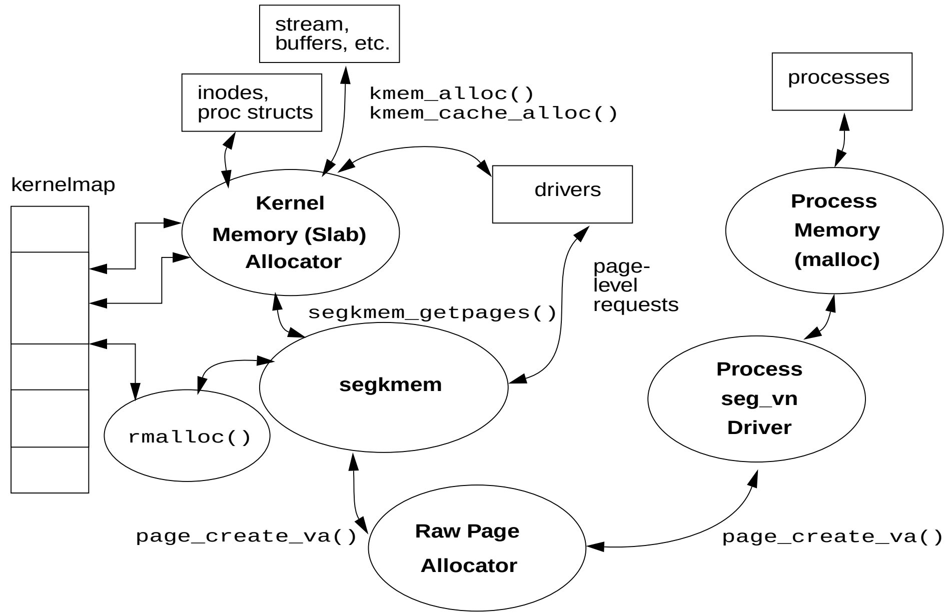
8-KByte Page



4-MByte Page



Kernel Memory



stream, buffers, etc.

inodes, proc structs

kmem_alloc()
kmem_cache_alloc()

processes

kernelmap

Kernel Memory (Slab) Allocator

drivers

Process Memory (malloc)

segkmem_getpages()

page-level requests

segkmem

rmalloc()

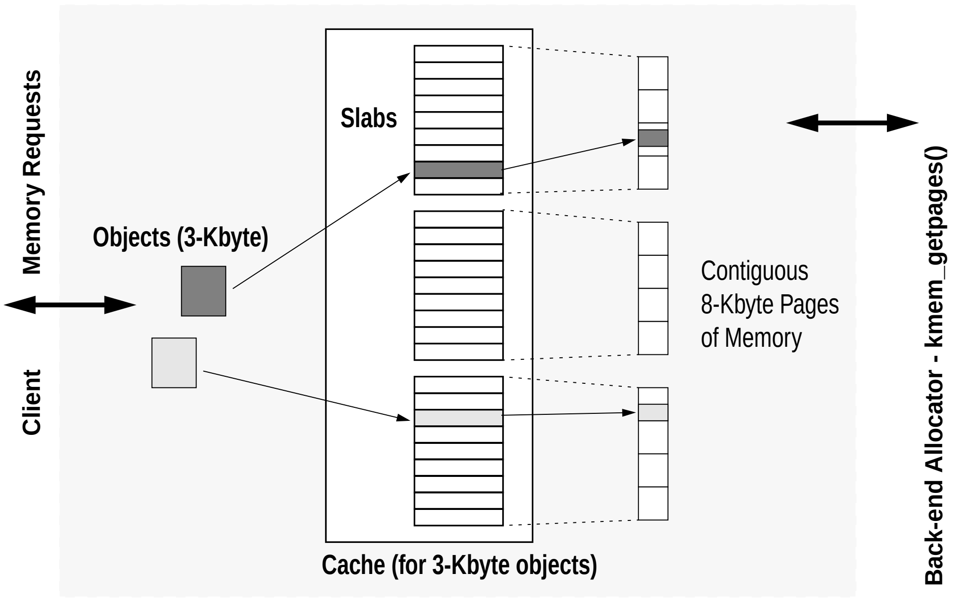
Process seg_vn Driver

page_create_va()

Raw Page Allocator

page_create_va()

Slab Allocator



Memory Requests

Client

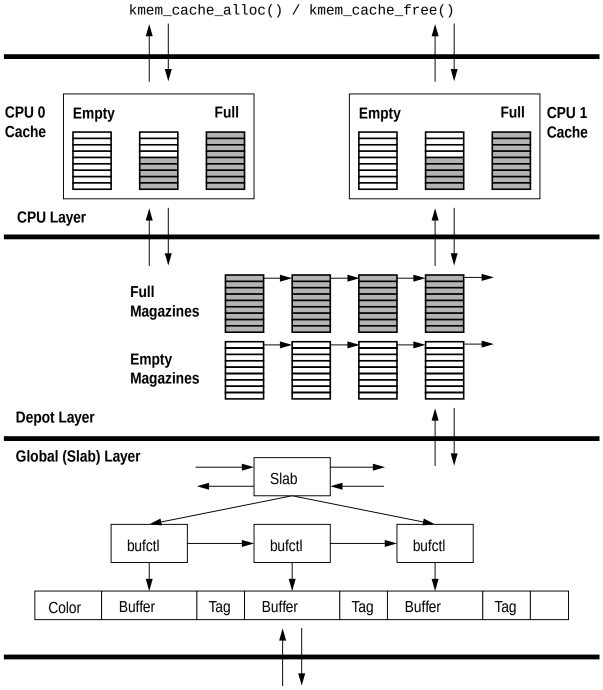
Objects (3-Kbyte)

Slabs

Contiguous 8-Kbyte Pages of Memory

Cache (for 3-Kbyte objects)

Back-end Allocator - kmem_getpages()



File Systems

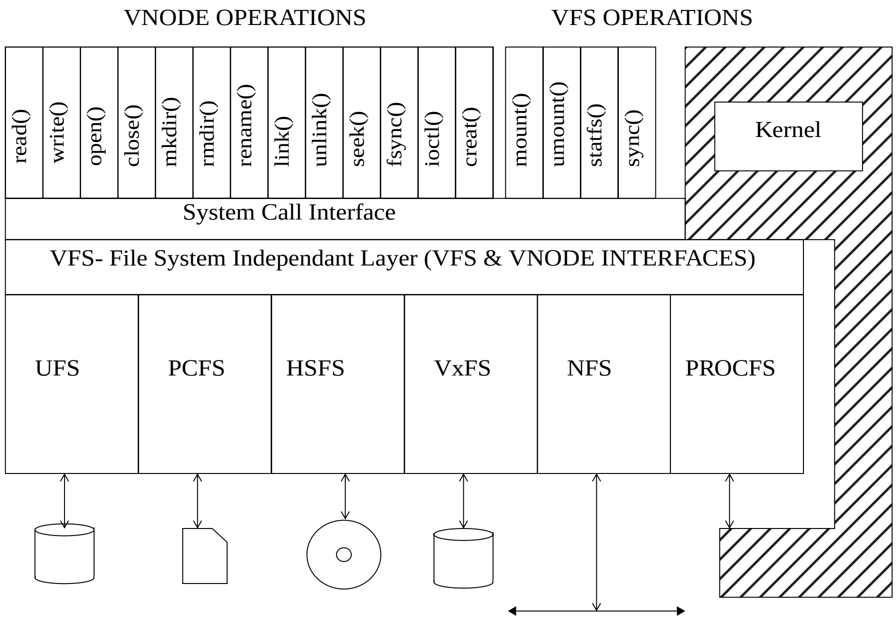
The File System Framework

- **SunOS was enhanced to support multiple file system types in 1985 to allow UFS & NFS**
  - UFS is the vnode implementation of BSD 4.2 FFS
  - Virtual file node was introduced - vnode
  - Virtual file system interface was introduced
- **File systems are modular**
  - Multiple Regular File Systems
  - Psuedo File Systems

File System Types

Filesystem	Type	Device	Description
ufs	Regular	Disk	Unix Fast Filesystem, default in Solaris
pcfs	Regular	Disk	MSDOS filesystem
hsfs	Regular	Disk	High Sierra File System (CDROM)
tmpfs	Regular	Memory	Uses memory and swap
nfs	Psuedo	Network	Network filesystem
cachefs	Psuedo	Filesystem	Uses a local disk as cache for another NFS file system
autofs	Psuedo	Filesystem	Uses a dynamic layout to mount other file systems
specfs	Psuedo	Device Drivers	Filesystem for the /dev devices
procfs	Psuedo	Kernel	/proc filesystem representing processes
sockfs	Psuedo	Network	Filesystem of socket connections
fifofs	Psuedo	Files	FIFO File System

The virtual file system framework



VNODE OPERATIONS

VFS OPERATIONS

read()
write()
open()
close()
mkdir()
rmdir()
rename()
link()
unlink()
seek()
fsync()
ioctl()
creat()

mount()
umount()
statfs()
sync()

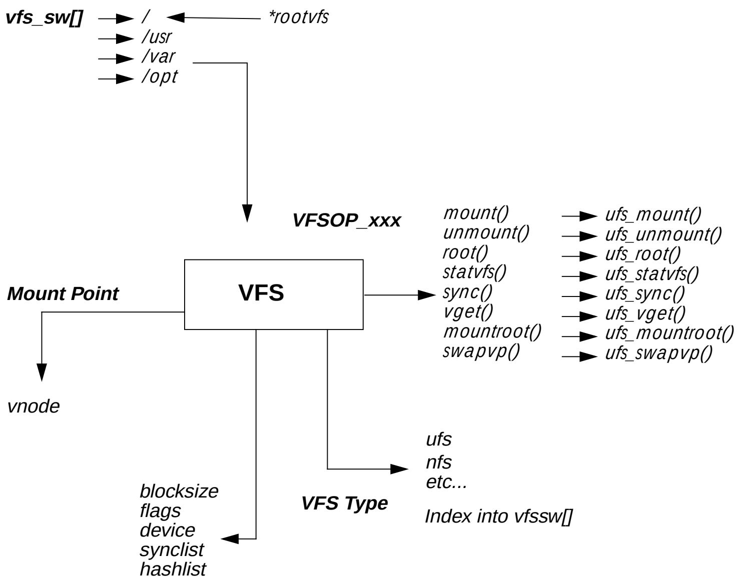
Kernel

System Call Interface

VFS- File System Independant Layer (VFS & VNODE INTERFACES)

UFS
PCFS
HSFS
VxFS
NFS
PROCFS

The VFS Interface



vfs_sw[]
/
/usr
/var
/opt
*rootvfs

VFSOP_xxx

mount()
unmount()
root()
statvfs()
sync()
vget()
mountroot()
swapvp()

ufs_mount()
ufs_unmount()
ufs_root()
ufs_statvfs()
ufs_sync()
ufs_vget()
ufs_mountroot()
ufs_swapvp()

Mount Point

VFS

vnode

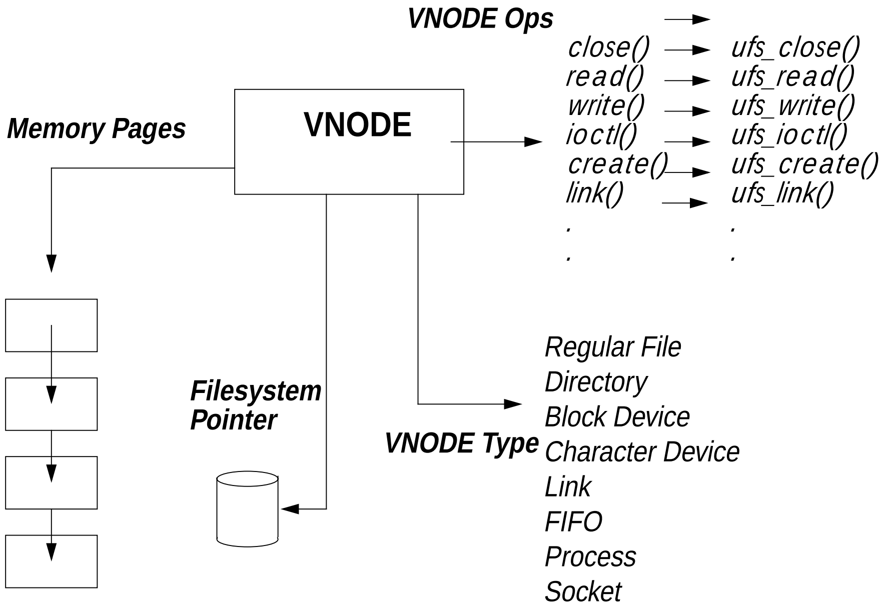
ufs
nfs
etc...

VFS Type

Index into vfssw[]

blocksize
flags
device
synclist
hashlist

The vnode interface



VNODE Ops

close() → ufs_close()
read() → ufs_read()
write() → ufs_write()
ioctl() → ufs_ioctl()
create() → ufs_create()
link() → ufs_link()
. .
. .

Memory Pages

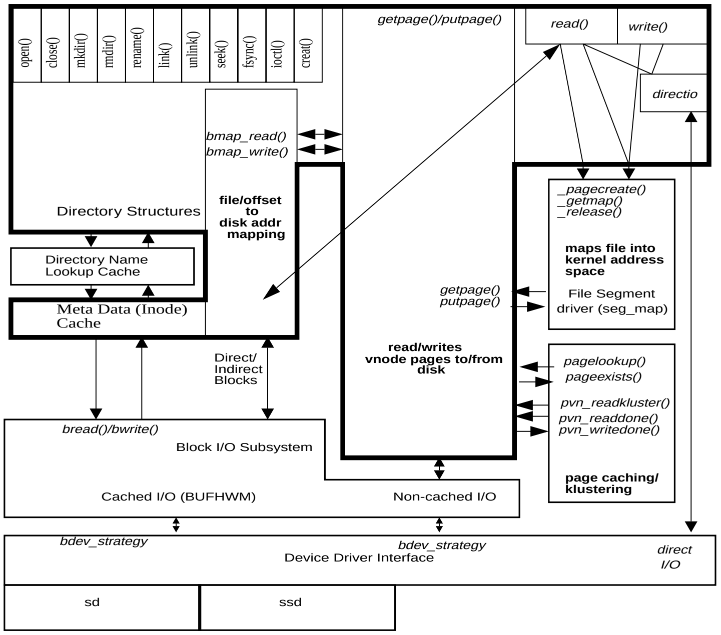
VNODE

Filesystem Pointer

VNODE Type

Regular File
Directory
Block Device
Character Device
Link
FIFO
Process
Socket

File System Architecture



open() close() mkdir() rmdir() rename() link() unlink() seek() fsync() ioctl() creat()

getpage()/putpage()

read() write()

directio

bmap_read()
bmap_write()

Directory Structures

file/offset to disk addr mapping

Directory Name Lookup Cache

Meta Data (Inode) Cache

_pagecreate()
_getmap()
_release()

maps file into kernel address space

File Segment driver (seg_map)

getpage()
putpage()

read/writes vnode pages to/from disk

pagelookup()
pageexists()

pvn_readkluster()
pvn_readdone()
pvn_writedone()

page caching/ klustering

Direct/ Indirect Blocks

bread()/bwrite()

Block I/O Subsystem

Cached I/O (BUFHWM)

Non-cached I/O

bdev_strategy

bdev_strategy

Device Driver Interface

direct I/O

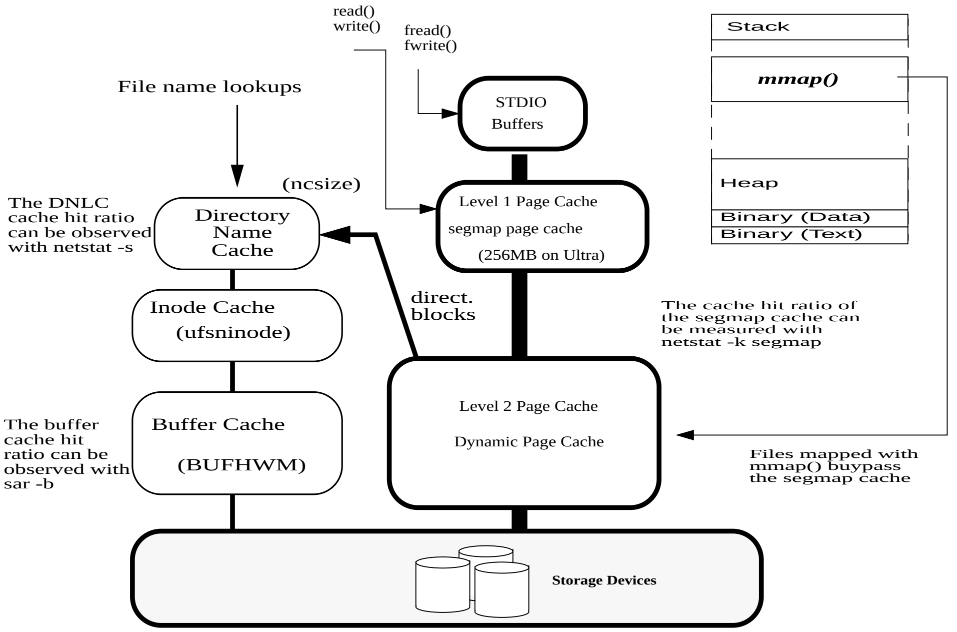
sd

ssd

File system Caching

- Solaris file systems use the VM system to cache and move data
- Regular reads are page ins, delayed writes are page outs
- VM Parameters and load dramatically effects file system performance
- Solaris 8 gives executable, stack and heap pages priority over file system pages

File System Caching



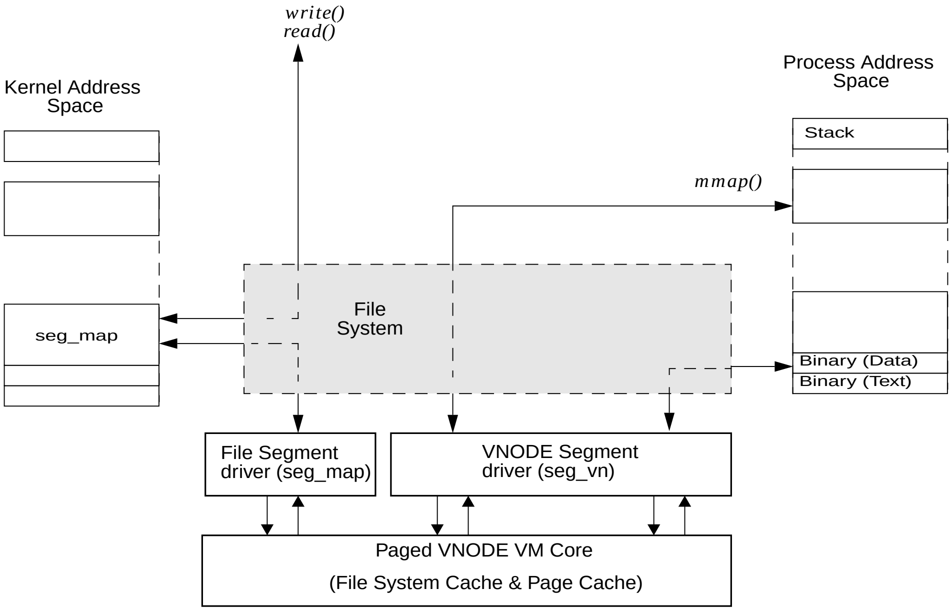
The DNLC cache hit ratio can be observed with netstat -s

The buffer cache hit ratio can be observed with sar -b

The cache hit ratio of the segmap cache can be measured with netstat -k segmap

Files mapped with mmap() buypass the segmap cache

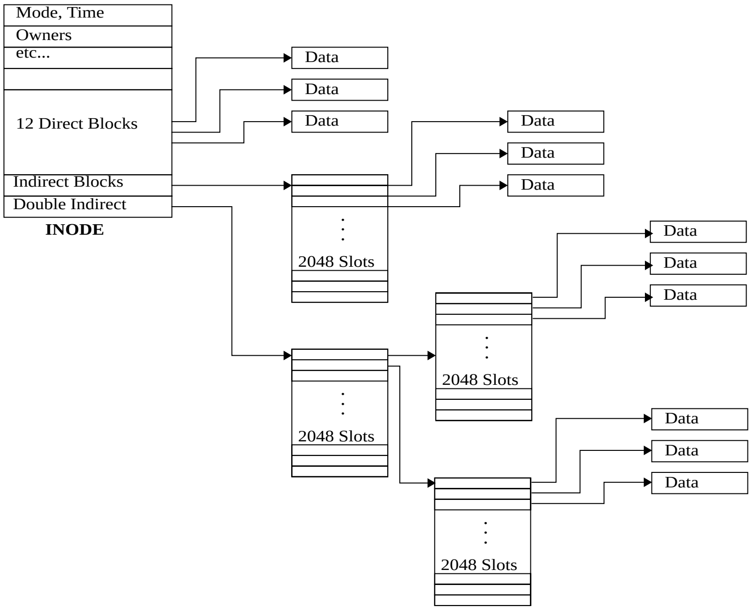
Segmap in more detail



UFS

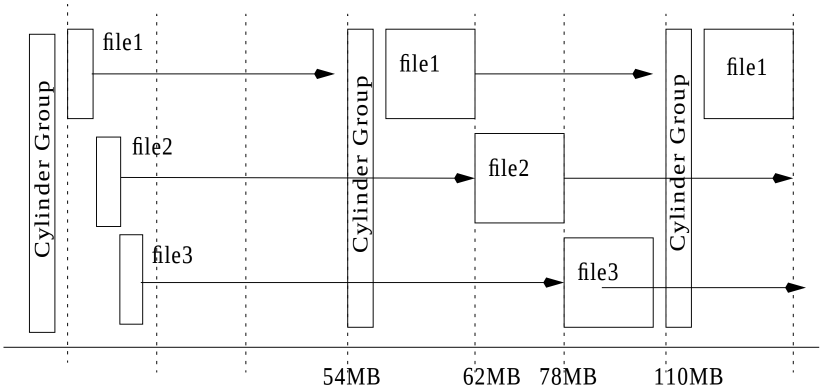
- **Block based allocation**
  - 2TB Max file system size
  - A file can grow to the max file system size
    - triple indirect is implemented
  - Prior to 2.6, max file size is 2GB

UFS On-Disk Layout



UFS Block Allocation

- **Allocation in cylinder groups, across the disk**
  - Blocks are allocated to the cylinder group starting at inode, until group has less than average free space
  - Allocation defaults to 16MB chunks



UFS Block Allocation

```
# filestat /home/bigfile

Inodes per cyl group:       64
Inodes per block:           64
Cylinder Group no:          0
Cylinder Group blk:         64
File System Block Size:     8192
Device block size:          512
Number of device blocks:    204928

Start Block    End Block    Length (Device Blocks)
-----------    -----------  ----------------------
      66272 -> 66463        192
      66480 -> 99247        32768
    1155904 -> 1188671      32768
    1277392 -> 1310159      32768
    1387552 -> 1420319      32768
    1497712 -> 1530479      32768
    1607872 -> 1640639      32768
    1718016 -> 1725999      7984
    1155872 -> 1155887      16

Number of extents:          9
Average extent size:        22769 Blocks
```

```
Note: The filestat command is show for demonstration purposes, and is not as yet
included with the Solaris operating system
```

UFS Logging

- **Beginning in Solaris 7, UFS logging became a mount option**
- **Log to spare blocks in the file system (no metadevice)**
- **Fast reboots - no fsck requires**

UFS Direct I/O

- **File systems cause a lot of paging activity**
- **Solaris 2.6 introduces a mechanism to bypass the VM system**
  - Forces completely unbuffered I/Os
  - Very slow writes (synchronous)
  - Useful for copying large files or when application does caching e.g. Oracle
  - mount -o forcedirectio /dev/xyz /mountpt
  - directio (fd, DIRECTIO_ON | DIRECTIO_OFF)

Direct I/O Checklist

- **Must be aligned**
  - sector aligned (512 byte boundary)
- **Must not be mapped**
- **Logging must be disabled**

UFS Write Throttle

- **A throttle exists in UFS to limit the amount of memory UFS can saturate, per file**
  - Controlled by three parameters
  - ufs_WRITES (1 = enabled)
  - ufs_HW = 393216 bytes (high water mark to suspend IO)
  - ufs_LW = 262144 bytes (low water mark to start IO)
- **Almost always need to set this higher to get maximum sequential write performance**
  - set ufs_LW=4194304
  - set ufs_HW=67108864

UFS Performance

- **Adjacent blocks are grouped and written together or read ahead**
  - Controlled by the maxcontig parameter
  - Defaults to 128k on most platforms, 1MB on SPARCstorage array 100,200
  - Must be set higher to achieve adequate write performance
  - maxphys must be raised beyond 128k also

The tmpfs file system

- **A fast hybrid disk/memory based file system**
  - mounted on /tmp by default
  - volatile across reboot
  - near zero disk latency
  - directory and meta-data in memory
- **File Data Blocks**
  - Looks just like process memory
  - Consumes memory from the free list!
  - Can be swapped out page at a time

The tmpfs file system

- **Can be mounted on other directories**
  - tmpfs can be mounted over existing directories
  - e.g. temporary file directory
- **Useful mount options**
  - can be limited in size -o size=
  - overlay mount option -O

```
# mount -F tmpfs -o size=100m swap /mytmp

# mount -F tmpfs -O -o size=100m swap /home/rmc/tmp
```

tmpfs Performance

- **Very fast write operations**
  - Writes to memory
  - file and directory creates to memory
- **Vast improvements in Solaris 2.6**
  - much faster directory operations
- **Limits**
  - 2GB max file system size pre 2.5
  - 2GB max file size without Solaris 7 64 bit mode
- **!! Priority Paging treats tmpfs as app. memory !!**

That’s About It...

- **There are a great many components and subsystems in the Solaris system**
- **We focused on the primary subsystems here; the things that are at the core of the kernel**

Thank You!

Tidbits, Tools & Techniques

The following pages are included as supplemental reference material for the student. It is not intended that this material will be covered during the course of the tutorial.

Kernel Organization

- **/kernel - platform independent components**
  - genunix - generic part of the core kernel
  - Subdirectories with various kernel modules
- **/platform - platform dependent components**
  - <platform_type> sundirectory (e.g. sun4u)
  - kernel - subdirectory with module subdirectories and platform specific unix (an optimized genunix on sun4u architectures only)
  - ufsboot - primary bootstrap code

Kernel Organization

- **/platform (continued)**
  - cprboot, cprbooter - checkpoint/resume from boot code
  - kadb - kernel debugger, supports “`boot kadb kernel/unix`”
- **/usr/kernel**
  - Additional kernel modules in the drv, exec, fs, misc, strmod, sys, sched subdirectories
  - Modules *not* required for core OS functions - generally loaded as a result of a application (e.g. RT scheduling class)

Kernel Organization

- **/usr/platform**
  - Platform specific objects not required in root filesystem
  - Binaries & header files
- **/devices - actual device special files**
  - Built from OpenBoot PROM device tree
- **/dev - symbolic links to actual device special files**
  - `devlinks(1M)` & `/etc/devlink.tab`

Kernel Organization

- **/opt - source directory for optional software packages**
  - Compilers, Volume Managers, etc
- **/etc - system administrative/control/config files**
  - /etc/system - kernel configuration control file
- **/var - logs, spool directories**
  - /var/sadm/system/[logs, data] - new locations for log files, etc

Kernel Organization

- **/bin & /usr/bin**
  - Shell commands (same directory)
- **/sbin - statically linked executables**
  - Availability of runtime linker not required
  - Startup stuff (init)
- **/proc**
  - procfs entry point
  - An “in-memory” pseudo file system

Pre S8 Caching Dynamics

- **UFS: when free memory is below *lotsfree + pages_before_pager***
  - UFS 8K reads and writes are subject to free-behind, all others are buffered
  - Read of sequential blocks are subject to free behind
- **Random I/O or non-8k I/O will cause the system to page heavily, stealing memory from applications.**

```
# vmstat 3
 procs     memory            page            disk          faults      cpu
 r b w   swap  free  re  mf pi po fr de sr m1 m2 s0 s2   in   sy   cs us sy id
 0 276 933 9046832 1996280 56 502 31660 2993 11496 0 1617 1301 154 186 0 1747 495 1198 0 8 92
 0 538 1240 7948848 103168 4 320 45298 2154 14040 0 1702 1986 0 330 0 2594 532 1705 0 8 92
 0 540 1240 7949104 100888 5 278 43072 2240 14650 0 1757 2000 0 280 0 2543 497 1652 0 9 91
 0 558 1240 7949136 102520 2 300 43016 1552 12986 0 1666 1932 0 293 0 2527 544 1684 0 8 92
 0 549 1240 7949112 98976 9 307 43442 2400 15696 0 1958 1998 0 322 0 2559 520 1659 0 8 92
 0 553 1240 7949088 104472 7 382 43264 2496 15061 0 1838 2020 0 392 0 2671 580 1865 0 8 92
 0 565 1240 7948952 104808 13 339 40944 2285 13480 0 1625 2046 0 328 0 2617 547 1752 0 7 93
 0 576 1240 7948896 101088 10 330 39888 1794 14074 0 1860 2061 0 326 0 2638 558 1720 0 8 92
 0 559 1240 7948944 100872 5 323 46274 1816 14037 0 1886 2053 0 333 0 2615 527 1773 0 8 92
 0 562 1240 7948936 96144 8 350 43362 2424 15834 0 2344 1999 0 363 0 2631 656 1767 0 9 91
```

Priority Paging

- **Pre-Solaris 8 *only***
  - Make sure it’s disabled in Solaris 8
- **Stops random or non-8K filesystem I/O from slowing the system**
- **Pager only frees application pages when there is a real memory shortage**
- **Useful for:**
  - Workstations with >64MB memory
  - OLTP workloads
  - Batch processing
  - Consolidated workloads

LRU Algorithm -

- **Without Priority Paging**

```
# memstat 3

memory  ---------- paging ----------- - executable -  - anonymous -  -- filesys -- --- cpu ---
  free  re  mf   pi   po   fr  de   sr  epi  epo  epf  api  apo  apf  fpi  fpo  fpf us sy wt id
 21160   0  22    0    5    5   0   0    0    0    0    0    0    0    0    5    5  0  1  0 99
 21152   0   0    0    0    0   0   0    0    0    0    0    0    0    0    0    0  0  0  0 100
 21152   0  18   34    2    2   0   0    0    0    0    0    0    0   34    2    2  0  1  0 99
 11920   0   0  277  106  272   0 153    0    0   32    0   98  149  277    8   90  0  3  0 97
 11888   0   0  256   69  224   0 106    0    0   16    0   69  178  256    0   29  0  3  1 96
 11896   0   0  213  106  261   0 124    0    0   26    0  106  232  213    0    2  0  3 13 84
 11904   0   0  245   66  242   0 122    0    0   16    0   64  221  245    2    5  0  2  0 98
 11896   0   0  245   64  224   0 132    0    0   21    0   64  189  245    0   13  0  2  0 98
```

- **With Priority Paging**

```
# memstat 3

memory  ---------- paging ----------- - executable -  - anonymous -  -- filesys -- --- cpu ---
  free  re  mf   pi   po   fr  de   sr  epi  epo  epf  api  apo  apf  fpi  fpo  fpf us sy wt id
 21160   0  22    0    5    5   0   0    0    0    0    0    0    0    0    5    5  0  1  0 99
 21152   0   0    0    0    0   0   0    0    0    0    0    0    0    0    0    0  0  0  0 100
 21152   0  18   34    2    2   0   0    0    0    0    0    0    0   34    2    2  0  1  0 99
 11920   0   0  277    8  272   0 153    0    0    0    0    0    0  277    8  272  0  3  0 97
 11888   0   0  256    0  224   0 106    0    0    0    0    0    0  256    0  224  0  3  1 96
 11896   0   0  213    0  261   0 124    0    0    0    0    0    0  213    0  261  0  3 13 84
 11904   0   0  245    2  242   0 122    0    0    0    0    0    0  245    2  242  0  2  0 98
 11896   0   0  245    0  224   0 132    0    0    0    0    0    0  245    0  224  0  2  0 98
```

Priority Paging

- **Solaris 2.7 FCS or Solaris 2.6 with T-105181-09**
  - http://devnull.eng/rmc/priority_paging.html
  - Set priority_paging=1 in /etc/system
- **Solaris 2.7 Extended vmstat**
  - ftp://playground.sun.com/pub/rmc/memstat

File System Tuning

- **set maxcontig to size of stripe width, e.g. 10 disks with 256k interleave = 2560k = 320blks**

```
 # newfs -C 320
```

- **Allow SCSI transfers up to 8MB in the IO, Disksuite and VxVM layers:**

```
set maxphys=8388608
set md_maxphys=8388608
set vxio:vol_maxio=16384
```

- **set the write throttle higher for large systems > 1GB of memory**

```
set ufs_LW=4194304
set ufs_HW=67108864
```

- **Increase maxpgio to prevent the page scanner from limiting writes**

```
set maxpgio=65536
```

- **Increase fastscan to limit the effect the page scanner has on file system thoughput**

```
set fastscan=65536
```

- **Enable Priority Paging**

```
set priority_paging=1
```

- **If using RAID5, ensure that alignment is set where possible**

```
# mkfs -F vxfs -o bsize=8192,align=320
```

- **If building temporary files, turn on fast, unsafe mode with fastfs (from Solaris install CD)**

```
# fastfs -f /filesys (on)
# fastfs -s /filesys (off)
```

- **If filesystems have thousands of files, increase the directory and inode caches**

```
set ncsize=32768  (keep 32k file names in the name cache)
set ufs_ninode=65536 (keep 64k inode structures in the inode cache)
set vxfs_ninode=65536 (keep 64k VxFS inode structures in the inode cache)
```

Large Files

- **Solaris 2.6 added support for large files**
  - In conformance with the large file summit API’s
  - Support for 64 bit offsets on 32 bit platforms
  - UFS supports large files (1TB)
  - Commands enhanced to deal with large files
  - man largefile(5)
- **Solaris 2.6 Large File Application Environment**
  - man lfcompile(5) lfcompile64(5)
  - Compile with _FILE_OFFSET_BITS=64
- **Solaris 2.7 Large Files**
  - 32 bit environment the same as Solaris 2.6
  - 64 bit environment has large file support by default
  - off_t is 64 bits

Tracing

- **Trace user signals and system calls - truss**
  - Traces by stopping and starting the process
  - Can trace system calls, inline or as a summary
  - Can also trace shared libraries and a.out
- **Linker/library interposing/profiling/tracing**
  - LD_ environment variables enable link debugging
  - man ld.so.1
  - using the LD_PRELOAD env variable
- **Trace Normal Formal (TNF)**
  - Kernel and Process Tracing
  - Lock Tracing
- **Kernel Tracing**
  - lockstat, tnf, kgmon

Process Tracing - truss

```
# truss -d dd if=500m of=/dev/null bs=16k count=2k 2>&1 |more
Base time stamp:  925931550.0927  [ Wed May  5 12:12:30 PDT 1999 ]
 0.0000 execve("/usr/bin/dd", 0xFFBEF68C, 0xFFBEF6A4)  argc = 5
 0.0034 open("/dev/zero", O_RDONLY)                      = 3
 0.0039 mmap(0x00000000, 8192, PROT_READ|PROT_WRITE|PROT_EXEC, MAP_PRIVATE, 3, 0) = 0xFF3A0000
 0.0043 open("/usr/lib/libc.so.1", O_RDONLY)             = 4
 0.0047 fstat(4, 0xFFBEF224)                             = 0
 0.0049 mmap(0x00000000, 8192, PROT_READ|PROT_EXEC, MAP_PRIVATE, 4, 0) = 0xFF390000
 0.0051 mmap(0x00000000, 761856, PROT_READ|PROT_EXEC, MAP_PRIVATE, 4, 0) = 0xFF280000
 0.0054 munmap(0xFF324000, 57344)                        = 0
 0.0057 mmap(0xFF332000, 25284, PROT_READ|PROT_WRITE|PROT_EXEC, MAP_PRIVATE|MAP_FIXED, 4, 663552) = 0xFF332000
 0.0062 close(4)                                         = 0
 0.0065 open("/usr/lib/libdl.so.1", O_RDONLY)            = 4
 0.0068 fstat(4, 0xFFBEF224)                             = 0
 0.0070 mmap(0xFF390000, 8192, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED, 4, 0) = 0xFF390000
 0.0073 close(4)                                         = 0
 0.0076 open("/usr/platform/SUNW,Ultra-2/lib/libc_psr.so.1", O_RDONLY) = 4
 0.0079 fstat(4, 0xFFBEF004)                             = 0
 0.0082 mmap(0x00000000, 8192, PROT_READ|PROT_EXEC, MAP_PRIVATE, 4, 0) = 0xFF380000
 0.0084 mmap(0x00000000, 16384, PROT_READ|PROT_EXEC, MAP_PRIVATE, 4, 0) = 0xFF370000
 0.0087 close(4)                                         = 0
 0.0100 close(3)                                         = 0
 0.0103 munmap(0xFF380000, 8192)                         = 0
 0.0110 open64("500m", O_RDONLY)                         = 3
 0.0115 creat64("/dev/null", 0666)                       = 4
 0.0119 sysconfig(_CONFIG_PAGESIZE)                      = 8192
 0.0121 brk(0x00023F40)                                  = 0
 0.0123 brk(0x0002BF40)                                  = 0
 0.0127 sigaction(SIGINT, 0xFFBEF470, 0xFFBEF4F0)        = 0
 0.0129 sigaction(SIGINT, 0xFFBEF470, 0xFFBEF4F0)        = 0
 0.0134 read(3, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)     = 16384
 0.0137 write(4, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)    = 16384
 0.0140 read(3, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)     = 16384
 0.0143 write(4, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)    = 16384
 0.0146 read(3, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)     = 16384
 0.0149 write(4, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)    = 16384
 0.0152 read(3, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)     = 16384
 0.0154 write(4, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)    = 16384
 0.0158 read(3, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)     = 16384
 0.0160 write(4, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)    = 16384
```

Tracing only specific System Calls

```
# truss -d -t read -t write dd if=500m of=/dev/null bs=16k count=2k
Base time stamp:  925931672.4494  [ Wed May  5 12:14:32 PDT 1999 ]
 0.0087 read(3, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)    = 16384
 0.0091 write(4, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)   = 16384
 0.0096 read(3, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)    = 16384
 0.0099 write(4, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)   = 16384
 0.0103 read(3, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)    = 16384
 0.0106 write(4, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)   = 16384
 0.0111 read(3, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)    = 16384
 0.0114 write(4, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)   = 16384
 0.0118 read(3, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)    = 16384
 0.0121 write(4, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)   = 16384
 0.0127 read(3, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)    = 16384
 0.0129 write(4, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)   = 16384
 0.0135 read(3, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)    = 16384
 0.0137 write(4, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)   = 16384
 0.0142 read(3, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)    = 16384
 0.0145 write(4, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)   = 16384
 0.0150 read(3, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)    = 16384
 0.0153 write(4, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)   = 16384
 0.0158 read(3, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)    = 16384
 0.0161 write(4, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)   = 16384
 0.0166 read(3, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)    = 16384
 0.0169 write(4, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)   = 16384
 0.0174 read(3, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)    = 16384
 0.0177 write(4, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)   = 16384
 0.0182 read(3, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)    = 16384
 0.0185 write(4, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)   = 16384
 0.0190 read(3, "\0\0\0\0\0\0\0\0\0\0\0\0".., 16384)    = 16384
```

System Call Summary - truss

- **Counts total cpu seconds per system call and calls**

```
# truss -c dd if=500m of=/dev/null bs=16k count=2k

syscall      seconds   calls  errors
_exit            .00       1
read             .34    2048
write            .03    2056
open             .00       4
close            .00       6
brk              .00       2
fstat            .00       3
execve           .00       1
sigaction        .00       2
mmap             .00       7
munmap           .00       2
sysconfig        .00       1
llseek           .00       1
creat64          .00       1
open64           .00       1
                ----     ---    ---
sys totals:      .37    4136      0
usr time:        .00
elapsed:         .89
```

Library Tracing - truss -u

```
# truss -d -u a.out,libc dd if=500m of=/dev/null bs=16k count=2k
Base time stamp:  925932005.2498  [ Wed May  5 12:20:05 PDT 1999 ]
 0.0000 execve("/usr/bin/dd", 0xFFBEF68C, 0xFFBEF6A4)  argc = 5
 0.0073 open("/dev/zero", O_RDONLY)                     = 3
 0.0077 mmap(0x00000000, 8192, PROT_READ|PROT_WRITE|PROT_EXEC, MAP_PRIVATE, 3, 0) = 0xFF3A0000
 0.0094 open("/usr/lib/libc.so.1", O_RDONLY)            = 4
 0.0097 fstat(4, 0xFFBEF224)                            = 0
 0.0100 mmap(0x00000000, 8192, PROT_READ|PROT_EXEC, MAP_PRIVATE, 4, 0) = 0xFF390000
 0.0102 mmap(0x00000000, 761856, PROT_READ|PROT_EXEC, MAP_PRIVATE, 4, 0) = 0xFF280000
 0.0105 munmap(0xFF324000, 57344)                       = 0
 0.0107 mmap(0xFF332000, 25284, PROT_READ|PROT_WRITE|PROT_EXEC, MAP_PRIVATE|MAP_FIXED, 4, 663552) =
0xFF332000
 0.0113 close(4)                                        = 0
 0.0116 open("/usr/lib/libdl.so.1", O_RDONLY)           = 4
 0.0119 fstat(4, 0xFFBEF224)                            = 0
 0.0121 mmap(0xFF390000, 8192, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_FIXED, 4, 0) = 0xFF390000
 0.0124 close(4)                                        = 0
 0.0127 open("/usr/platform/SUNW,Ultra-2/lib/libc_psr.so.1", O_RDONLY) = 4
 0.0131 fstat(4, 0xFFBEF004)                            = 0
 0.0133 mmap(0x00000000, 8192, PROT_READ|PROT_EXEC, MAP_PRIVATE, 4, 0) = 0xFF380000
 0.0135 mmap(0x00000000, 16384, PROT_READ|PROT_EXEC, MAP_PRIVATE, 4, 0) = 0xFF370000
 0.0138 close(4)                                        = 0
 0.2369 close(3)                                        = 0
 0.2372 munmap(0xFF380000, 8192)                        = 0
 0.2380 -> libc:atexit(0xff3b9e8c, 0x23400, 0x0, 0x0)
 0.2398 <- libc:atexit() = 0
 0.2403 -> libc:atexit(0x12ed4, 0xff3b9e8c, 0xff334518, 0xff332018)
 0.2419 <- libc:atexit() = 0
 0.2424 -> _init(0x0, 0x12ed4, 0xff334518, 0xff332018)
 0.2431 <- _init() = 0
 0.2436 -> main(0x5, 0xffbef68c, 0xffbef6a4, 0x23400)
 0.2443   -> libc:setlocale(0x6, 0x12f14, 0x0, 0x0)
 0.2585   <- libc:setlocale() = 0xff31f316
 0.2590   -> libc:textdomain(0x12f18, 0x12f14, 0x0, 0xff335938)
 0.2617   <- libc:textdomain() = 0xff3359d8
 0.2622   -> libc:getopt(0x5, 0xffbef68c, 0x12f28, 0xff335938)
 0.2630   <- libc:getopt() = -1
```

Library Tracing - apptrace(1)

```
sunsys> apptrace ls
ls       -> libc.so.1:atexit(func = 0xff3caa24) = 0x0
ls       -> libc.so.1:atexit(func = 0x13ad4) = 0x0
ls       -> libc.so.1:setlocale(category = 0x6, locale = “”) = “/en_US.ISO8859-1/en_”
ls       -> libc.so.1:textdomain(domainname = “SUNW_OST_OSCMD”) = “SUNW_OST_OSCMD”
ls       -> libc.so.1:time(tloc = 0x0) = 0x3aee2678
ls       -> libc.so.1:isatty(fildes = 0x1) = 0x1
ls       -> libc.so.1:getopt(argc = 0x1, argv = 0xffbeeff4, optstring =
“RaAdC1xmnlogrtucpFbq”) = 0xffffffff errno = 0 (Error 0)
ls       -> libc.so.1:getenv(name = “COLUMNS”) = “<nil>”
ls       -> libc.so.1:ioctl(0x1, 0x5468, 0x2472a)
ls       -> libc.so.1:malloc(size = 0x100) = 0x25d10
ls       -> libc.so.1:malloc(size = 0x9000) = 0x25e18
ls       -> libc.so.1:lstat64(path = “.”, buf = 0xffbeee98) = 0x0
ls       -> libc.so.1:qsort(base = 0x25d10, nel = 0x1, width = 0x4, compar = 0x134bc)
ls       -> libc.so.1:.div(0x50, 0x3, 0x50)
ls       -> libc.so.1:.div(0xffffffff, 0x1a, 0x0)
ls       -> libc.so.1:.mul(0x1, 0x0, 0xffffffff)
ls       -> libc.so.1:.mul(0x1, 0x1, 0x0)

[snip]

ls       -> libc.so.1:.mul(0x1, 0xf, 0x0)
ls       -> libc.so.1:.mul(0x1, 0x10, 0x0)
ls       -> libc.so.1:.mul(0x1, 0x11, 0x0)
ls       -> libc.so.1:.mul(0x1, 0x12, 0x0)
ls       -> libc.so.1:.mul(0x1, 0x13, 0x0)
ls       -> libc.so.1:.mul(0x1, 0x14, 0x0)
ls       -> libc.so.1:.mul(0x1, 0x15, 0x0)
ls       -> libc.so.1:.mul(0x1, 0x16, 0x0)
ls       -> libc.so.1:.mul(0x1, 0x17, 0x0)
ls       -> libc.so.1:.mul(0x1, 0x18, 0x0)
ls       -> libc.so.1:.mul(0x1, 0x19, 0x0)
ls       -> libc.so.1:opendir(dirname = “.”) = 0x2ee20
ls       -> libc.so.1:readdir64(dirp = 0x2ee20) = 0x2ee30
ls       -> libc.so.1:readdir64(dirp = 0x2ee20) = 0x2ee48
ls       -> libc.so.1:readdir64(dirp = 0x2ee20) = 0x2ee60
ls       -> libc.so.1:readdir64(dirp = 0x2ee20) = 0x2ee80
ls       -> libc.so.1:readdir64(dirp = 0x2ee20) = 0x2eea0
```

Library Tracing - LD_PROFILE

```
# export LD_PROFILE=/usr/lib/libc.so.1
# ls
# gprof /usr/lib/libc.so.1 /var/tmp/libc.so.1.profile

  %  cumulative    self              self    total
 time   seconds   seconds    calls  ms/call  ms/call name
 22.2       0.06     0.06   382786     0.00     0.00  fread [3]
 18.5       0.11     0.05       13     3.85     3.85  qsort [4]
  7.4       0.13     0.02     1356     0.01     0.02  printf [5]
  7.4       0.15     0.02      528     0.04     0.04  _libc_write [9]
  7.4       0.17     0.02      113     0.18     0.18  _read [10]
  7.4       0.19     0.02                            _set_orientation_byte [11]
  7.4       0.21     0.02                            qst [8]
  3.7       0.22     0.01     9794     0.00     0.00  _realbufend [19]
  3.7       0.23     0.01     9302     0.00     0.00  _mutex_held [14]
  3.7       0.24     0.01     2820     0.00     0.00  _strcoll [15]
  3.7       0.25     0.01       30     0.33     0.33  _ioctl [16]
  3.7       0.26     0.01                            __open [17]
  3.7       0.27     0.01                            _brk_unlocked [18]
  0.0       0.27     0.00    79809     0.00     0.00  strcmp [30]
  0.0       0.27     0.00     9302     0.00     0.00  _rw_read_held [723]
  0.0       0.27     0.00     7786     0.00     0.00  __flsbuf [12]
  0.0       0.27     0.00     3265     0.00     0.00  ..mul [31]
  0.0       0.27     0.00     2724     0.00     0.00  ..rem [32]
  0.0       0.27     0.00     1727     0.00     0.00  ..udiv [33]
  0.0       0.27     0.00     1641     0.00     0.00  ..umul [34]
  0.0       0.27     0.00     1458     0.00     0.01  _doprnt [13]
  0.0       0.27     0.00     1356     0.00     0.00  memchr [35]
  0.0       0.27     0.00     1176     0.00     0.00  _iswprint [724]
```

- **See “Linker and Libraries Guide”**
  - http://docs.sun.com

Library Tracing - LD_DEBUG

```
# export LD_DEBUG=help
# ls
00000:           For debugging the runtime linking of an application:
00000:                  LD_DEBUG=token1,token2  prog
00000:           enables diagnostics to the stderr.  The additional option:
00000:                  LD_DEBUG_OUTPUT=file
00000:           redirects the diagnostics to an output file created using
00000:           the specified name and the process id as a suffix.  All
00000:           diagnostics are prepended with the process id.
00000:
00000:
00000: args      display input argument processing (ld only)
00000: basic     provide basic trace information/warnings
00000: bindings  display symbol binding; detail flag shows absolute:relative
00000:           addresses (ld.so.1 only)
00000: detail    provide more information in conjunction with other options
00000: entry     display entrance criteria descriptors (ld only)
00000: files     display input file processing (files and libraries)
00000: help      display this help message
00000: libs      display library search paths; detail flag shows actual
00000:           library lookup (-l) processing
00000: map       display map file processing (ld only)
00000: move      display move section processing
00000: reloc     display relocation processing
00000: sections  display input section processing (ld only)
00000: segments  display available output segments and address/offset
00000:           processing; detail flag shows associated sections (ld only)
00000: support   display support library processing (ld only)
00000: symbols   display symbol table processing;
00000:           detail flag shows resolution and linker table addition
00000: versions  display version processing
00000: audit     display rt-link audit processing
00000: got       display GOT symbol information (ld only)
```

Library Tracing - LD_DEBUG

```
# export LD_DEBUG=basic
# ls

03617: cyclic objects for .init (Befor sorting)
03617:  /usr/lib/libc.so.1 IDX=3
03617:  /usr/lib/libmapmalloc.so.1 IDX=2
03617:  /usr/lib/link_audit/ldprof.so.1 IDX=1
03617: cyclic objects for .init (After sorting)
03617:  /usr/lib/libc.so.1 IDX=3
03617:  /usr/lib/libmapmalloc.so.1 IDX=2
03617:  /usr/lib/link_audit/ldprof.so.1 IDX=1
03617:
03617: calling init: /usr/lib/libc.so.1
03617:
03617:
03617: calling init: /usr/lib/libmapmalloc.so.1
03617:
03617:
03617: calling init: /usr/lib/link_audit/ldprof.so.1
03617:
03617:
03617: calling init: /usr/lib/libc.so.1
03617:
03617:
03617: transferring control: ls
03617:
AcrobatDistiller              crash                         options.el
AdobePhotoshop3               docs                          pc
Mail                          fmdictionary                  projects
Netscape                      fmfilesvisited                rm
Netscape.extra                gp2                           sarahdinner.htm
03617: calling fini: /usr/lib/libc.so.1
03617:
03617: cyclic objects for .fini (Befor sorting)
03617:  /usr/lib/libc.so.1 IDX=3
03617:  /usr/lib/liblddbg.so.4 IDX=2
03617: cyclic objects for .fini (After sorting)
03617:  /usr/lib/liblddbg.so.4 IDX=2
03617:  /usr/lib/libc.so.1 IDX=3
03617:
```

```
03617: calling fini: /usr/lib/liblddbg.so.4
03617:
03617:
03617: calling fini: /usr/lib/libc.so.1
03617:
03617: cyclic objects for .fini (Befor sorting)
03617:  /usr/lib/libmapmalloc.so.1 IDX=2
03617:  /usr/lib/libc.so.1 IDX=3
03617: cyclic objects for .fini (After sorting)
03617:  /usr/lib/libmapmalloc.so.1 IDX=2
03617:  /usr/lib/libc.so.1 IDX=3
03617:
03617: calling fini: /usr/lib/link_audit/ldprof.so.1
03617:
03617:
03617: calling fini: /usr/lib/libmapmalloc.so.1
03617:
03617:
03617: calling fini: /usr/lib/libc.so.1
03617:
```

- **To see everything:**

```
# export LD_DEBUG=args,bindings,detail,entry,files,libs,map,move,reloc,sections,seg-
ments,support,symbols
# ls

<lots of stuff>
```

Mike Bennets Tools

- **Interposing tools:**
  - Heap Library Tool - Interposes malloc() and free() calls to check for memory leaks
  - Lock Library Tool - Interposes mutex_lock() and mutex_unlock() to measure the time spent in locks
  - No code modifications; Tools work with binaries
- **Sampling Tools:**
  - thrstack - pstack-like program; prints the traceback of all user-level threads or LWPs in a process
  - thrprof - prof-like function; output is sorted by frequences of user-level thread stacks
  - No code modifications; Tools work with binaries
- **http://www.netwiz.net/~mbennett**

Library Interposing

- **A c-function can be inserted inline of a shared library function**
    - use LD_PRELOAD to load your function ahead of the real one

```
# cc -K PIC -c interpose_hostid.c
# ld -G -o interpose_hostid.so interpose_hostid.o
# export LD_PRELOAD=/home/rmc/lib/interpose_hostid.so

static int (*libnsl_gethostid)();

_init()
{
        struct stat buf;
        char str[1024];

        libnsl_gethostid = (int(*)())dlsym(RTLD_NEXT, "_gethostid");
        (void) printf(“library loaded!\n”);

}

int _gethostid(void)
{
        int result=0x0550040c0;
        int proper_hostid;

        proper_hostid=(*libnsl_gethostid)(void);
        (void) printf(“hostid() returns %d instead of %d\n”, result, proper_hostid);
        return result;
}
```

IO Tracing - etruss

```
$ etruss -c myprog

syscall       cpu sec  elapsed     latency     wait   calls  errors
read             .396    4.964     .248237      77%      20
write            .488     .514     .025725       0%      20
open             .001     .001     .000142       0%      11      6
close            .000     .000     .000057       0%       5
brk              .000     .000     .000060       0%       4
fstat            .000     .000     .000051       0%       4
sysconfig        .000     .000     .000042       0%       1
creat64          .027     .027     .027119       0%       1
open64           .000     .000     .000125       0%       1
                -----    -----                          ---    ---
sys totals:      .915    5.509                           84      6
usr time:        .004
elapsed:        5.870

                     ---------- per I/O ----------    ---------- total -------------
syscall[file]          latency      iowait      size     wait      Kb/s   calls
 read[3]               .248237     .228038   1048576      77%      3489      20
write[3]               .000000     .000000         0       0%         0       0
 read[5]               .000000     .000000         0       0%         0       0
write[5]               .025725     .000000   1048576       0%      3489      20

The etruss utility can be obtained from ftp://playground.sun.com/pub/rmc

Note: etruss does not support multi-threaded processes.
```

Tracing with TNF

- **TNF - Trace Normal Form**
  - Can be used on user executables or the kernel
  - Traces to a buffer and then the buffer can be dumped
  - Obtrusive tracing, inserts code inline
  - Minimal Overhead
- **TNF commands bundled with Solaris**
  - prex - control tnf start/stop etc
  - tnfxtract - dump tnf buffer to a file
  - tnfdump - print tnf buffer in ascii format
- **Unbundled TNF Toolkit**
  - Available from the developer web site - http://soldc.sun.com
  - Package is SUNWtnftl, includes a GUI analysis tool (tnfview)

A TNF Example

- **Using the TNF Toolkit scripts:**

```
# tnftrace -m ls.tnf -k -i libc -c /bin/ls
# tnfdump ls.tnf

---------------- ---------------- ----- ----- ---------- --- ------------------------ ------------------------
    Elapsed (ms)       Delta (ms)   PID LWPID    TID     CPU Probe Name               Data / Description . . .
---------------- ---------------- ----- ----- ---------- --- ------------------------ ------------------------
        0.000000         0.000000  4761     1          1   - atexit_start             func: 0xff3b9d88
        0.294823         0.294823  4761     1          1   - atexit_end               return_value: 0
        0.301004         0.006181  4761     1          1   - atexit_start             func: 0x13d64
        0.304860         0.003856  4761     1          1   - atexit_end               return_value: 0
       42.968569        42.663709  4761     1          1   - setlocale_start          cat: 6 loc: ""
       43.348068         0.379499  4761     1          1   - getenv_start             name: 0xff2246dc
      43.419287        0.071219  4761     1         1   - getenv_end             return_value: 0x0       <NULL>
       43.424958         0.005671  4761     1          1   - getenv_start             name: 0xff2246e4
      43.434163        0.009205  4761     1         1   - getenv_end             return_value: 0x0       <NULL>
       43.438457         0.004294  4761     1          1   - getenv_start             name: 0xff224650
       43.444074         0.005617  4761     1          1   - getenv_end               return_value: "en_US"
      43.513071         0.068997  4761     1         1   - strcmp_start           s1: 0xffbefb6f s2: 0xff221dc8
       44.027438         0.514367  4761     1          1   - strcmp_end               return_value: 21
       44.105484         0.078046  4761     1          1   - strchr_start             sp: 0xffbefb6f c: 47
      44.145142        0.039658  4761     1         1   - strchr_end             return_value: 0x0       <NULL>
       44.150939         0.005797  4761     1          1   - getenv_start             name: 0xff22465c
       44.160425         0.009486  4761     1          1   - getenv_end               return_value: "en_US"
      44.167791         0.007366  4761     1         1   - strcmp_start           s1: 0xffbefba0 s2: 0xff221dc8
       44.173630         0.005839  4761     1          1   - strcmp_end               return_value: 21
       44.177158         0.003528  4761     1          1   - strchr_start             sp: 0xffbefba0 c: 47
```

- **The GUI displays a timeline for each probe**

```
# tnfview ls.tnf
```

TNF Kernel Trace

- **Can Trace various pre-defined trace points in the kernel**

```
# tnftrace -m /tmp/kernel.tnf -k -t 5
# tndump /tmp/kernel.tnf

---------------- ---------------- ----- ----- ---------- --- ------------------------- ------------------------
    Elapsed (ms)       Delta (ms)   PID LWPID    TID     CPU Probe Name                Data / Description . . .
---------------- ---------------- ----- ----- ---------- --- ------------------------- ------------------------
     4668.561463         0.003301  4199     1 0x300        1 syscall_start             sysnum: 17 sys_name: "brk"
     4668.568721         0.007258  1286     1 0x300        0 syscall_end               rval1: 0 rval2: 2890520255360 errno: 0 errno_name: "NONE"
     4668.575275         0.006554  1286     1 0x300        0 syscall_start             sysnum: 100 sys_name: "context"
     4668.576988         0.001713  4199     1 0x300        1 syscall_end               rval1: 0 rval2: 4296179712 errno: 0 errno_name: "NONE"
     4668.582993         0.006005  4199     1 0x300        1 thread_state              state: 4 state_name: "DFAULT"
     4668.585821        0.002828  4199     1 0x300       1 address_fault           address: 0x1 fault_type: 0 access: 1 fault_type_name: "INVAL"
     4668.590291         0.004470  1286     1 0x300        0 syscall_end               rval1: 0 rval2: 2890520255360 errno: 0 errno_name: "NONE"
     4668.590909         0.000618  4199     1 0x300        1 anon_zero                 address: 0x1
     4668.596183         0.005274  1286     1 0x300        0 syscall_start             sysnum: 100 sys_name: "context"
     4668.611347         0.015164  1286     1 0x300        0 syscall_end               rval1: 0 rval2: 2890520255360 errno: 0 errno_name: "NONE"
     4668.621102         0.009755  1286     1 0x300        0 syscall_start             sysnum: 100 sys_name: "context"
     4668.636282         0.015180  1286     1 0x300        0 syscall_end               rval1: 0 rval2: 2890520255360 errno: 0 errno_name: "NONE"
     4668.642131         0.005849  1286     1 0x300        0 syscall_start             sysnum: 100 sys_name: "context"
     4668.657170         0.015039  1286     1 0x300        0 syscall_end               rval1: 0 rval2: 2890520255360 errno: 0 errno_name: "NONE"
     4668.658383         0.001213  4199     1 0x300        1 thread_state              state: 0 state_name: "USER"
     4668.664792         0.006409  1286     1 0x300        0 syscall_start             sysnum: 3 sys_name: "read"
     4668.675132         0.010340  1286     1 0x300        0 syscall_end               rval1: 11 rval2: 14033296 errno: 11 errno_name: "NONE"
     4668.683187         0.008055  1286     1 0x300        0 syscall_start             sysnum: 100 sys_name: "context"
     4668.684791         0.001604  4199     1 0x300        1 syscall_start             sysnum: 3 sys_name: "read"
     4668.699715         0.014924  1286     1 0x300        0 syscall_end               rval1: 0 rval2: 2890520255360 errno: 0 errno_name: "NONE"
     4668.705655         0.005940  1286     1 0x300        0 syscall_start             sysnum: 100 sys_name: "context"

# tnfview /tmp/kernel.tnf
```

Example: TNF IO Trace

- **Enable just the IO probes to get IO read/write/seek activity:**

```
# tnftrace -m /tmp/kernel.tnf -k -t 5 -e io
# tndump /tmp/kernel.tnf

---------------- ---------------- ----- ----- ---------- --- ------------------------- ------------------------
    Elapsed (ms)       Delta (ms)   PID LWPID    TID     CPU Probe Name                Data / Description . . .
---------------- ---------------- ----- ----- ---------- --- ------------------------- ------------------------
        0.000000         0.000000     0     0 0x2a1        1 strategy                  device: 5 block: 200896 size: 8192 buf: 0x300 flags: 72
flag_symbols: "B
_KERNBUF | B_READ"
        0.076590         0.076590     0     0 0x2a1        1 strategy                  device: 28 block: 100544 size: 8192 buf: 0x300 flags: 73
flag_symbols: "
B_BUSY | B_KERNBUF | B_READ"
        0.096149         0.019559     0     0 0x2a1        1 strategy                  device: 12 block: 100544 size: 8192 buf: 0x300 flags: 73
flag_symbols: "
B_BUSY | B_KERNBUF | B_READ"
       19.419294        19.323145     0     0 0x2a1        1 biodone                   device: 12 block: 100544 buf: 0x300
       19.428697         0.009403     0     0 0x2a1        1 biodone                   device: 28 block: 100544 buf: 0x300
       19.456649         0.027952     0     0 0x2a1        1 biodone                   device: 5 block: 200896 buf: 0x300
       19.515680         0.059031     0     0 0x2a1        1 biodone                   device: 5 block: 200896 buf: 0x300
       19.528553         0.012873     0     0 0x2a1        1 strategy                  device: 5 block: 1807968 size: 8192 buf: 0x300 flags: 72
flag_symbols: "
B_KERNBUF | B_READ"
       19.555603         0.027050     0     0 0x2a1        1 strategy                  device: 28 block: 904032 size: 8192 buf: 0x300 flags: 73
flag_symbols: "
B_BUSY | B_KERNBUF | B_READ"
       19.565500         0.009897     0     0 0x2a1        1 strategy                  device: 12 block: 904032 size: 8192 buf: 0x300 flags: 73
flag_symbols: "
B_BUSY | B_KERNBUF | B_READ"
       34.608626        15.043126     0     0 0x2a1        1 biodone                   device: 12 block: 904032 buf: 0x300
       34.613920         0.005294     0     0 0x2a1        1 biodone                   device: 28 block: 904032 buf: 0x300
       34.630514         0.016594     0     0 0x2a1        1 biodone                   device: 5 block: 1807968 buf: 0x300

# tnfview /tmp/kernel.tnf
```

Disabling the Console Break

- **A new feature was added in Solaris 2.6**
  - Man page was not updated until Solaris 7
  - Can be used to disable L1-A and the RS232 break
  - Useful to prevent the machine stopping when the console is power cycled
- **Use the kbd command to disable**
  - kbd -a disable
  - Set in /etc/default/kbd to make permanent

```
# KEYBOARD_ABORT affects the default behavior of the keyboard abort
# sequence, see kbd(1) for details.  The default value is "enable".
# The optional value is "disable".  Any other value is ignored.
#
# Uncomment the following lines to change the default values.
#
#KEYBOARD_ABORT=enable
```

Dump Configuration

- **Solaris 2.x -> 2.6 Dumps**
  - Only dumps kernel memory
  - Only requires about 15% of system memory size for dump
  - 2GB limit
  - Special configuration required for VxVM encapsulated disks
- **Solaris 8 Dumps**
  - New robust dump environment
  - Can dump kernel and/or user memory
  - 2G limit removed
  - New administration commands - dumpadm(1M)

```
example# dumpadm

        Dump content: kernel pages
         Dump device: /dev/dsk/c0t0d0s1 (swap)
  Savecore directory: /var/crash/saturn
    Savecore enabled: yes
```

Quick Tidbit

- **The Solaris FAQ of the decade - “*how many files can a process have open?*”**
- **If application uses stdio (fopen(3), fclose(3), etc)**
    - Limit is 255 max due to *fd* representation in FILE
    - Except in Solaris 7 & 8, 64-bit, then it’s 64k
- **If application uses select(3)**
    - FD_SETSIZE is 1k, and cannot be increased
    - Except in Solaris 7 & 8, for 64-bit it’s 64k
- **Otherwise, it’s resource limit dependent**

Quick Tip

- **Use /etc/crash(1M) to examine the “var” structure**

```
# /etc/crash
dumpfile = /dev/mem, namelist = /dev/ksyms, outfile = stdout
> v
v_buf: 100
v_call:   0
v_proc: 4058
v_nglobpris: 110
v_maxsyspri:  99
v_clist:   0
v_maxup: 4053
v_hbuf: 256
v_hmask: 255
v_pbuf:   0
v_sptmap:   0
v_maxpmem: 0
v_autoup: 30
v_bufhwm: 5196
> q
#
```

Quick Tip

- **sysdef(1M) works as well**

```
*
* Tunable Parameters
*
 5320704        maximum memory allowed in buffer cache (bufhwm)
    4058        maximum number of processes (v.v_proc)
      99        maximum global priority in sys class (MAXCLSYSPRI)
    4053        maximum processes per user id (v.v_maxup)
      30        auto update time limit in seconds (NAUTOUP)
      25        page stealing low water mark (GPGSLO)
       5        fsflush run rate (FSFLUSHR)
      25        minimum resident memory for avoiding deadlock (MINARMEM)
      25        minimum swapable memory for avoiding deadlock (MINASMEM)
*
```

- **For the hardcore “UNIX” fans...**

```
# adb -k /dev/ksyms /dev/mem
physmem fdde
ncsize/D
ncsize:
ncsize:         17564
ufs_ninode/D
ufs_ninode:
ufs_ninode:     17564
$q
#
```

Quick Tip

- **adb macros in /usr/lib/adb**
  - format is:

```
symbolic_reference$<macro, or address$<macro
```

```
# adb -k /dev/ksyms /dev/mem
physmem fdde
v$<v
v:
v:              buf             call            proc
                100             0               4058
v+0xc:          globpri         maxsyspri
                110             99
v+0x1c:         maxup           hbuf            hmask
                4053            256             ff
v+0x28:         pbuf            maxpmem         autoup
                0               0               30
v+0x38:         bufhwm
                5196
```

Thank You!