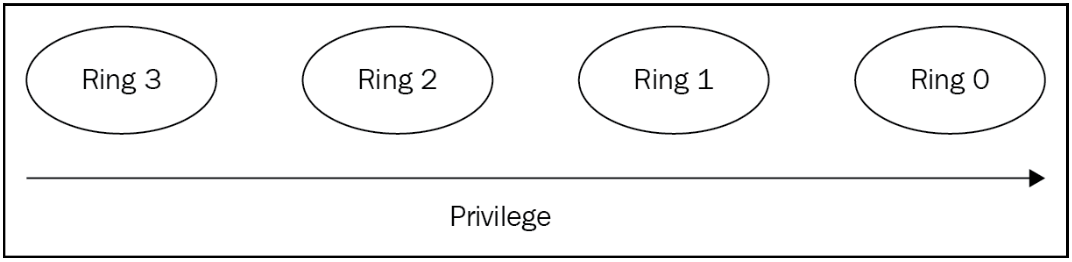
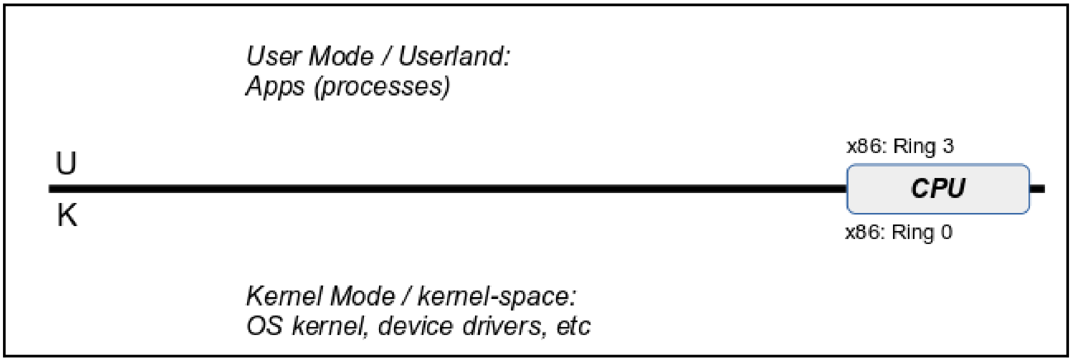
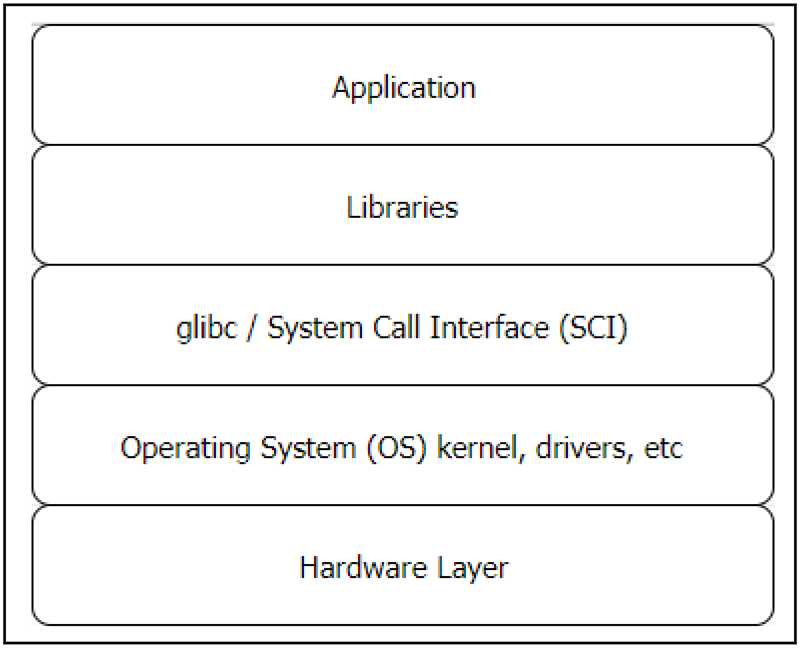
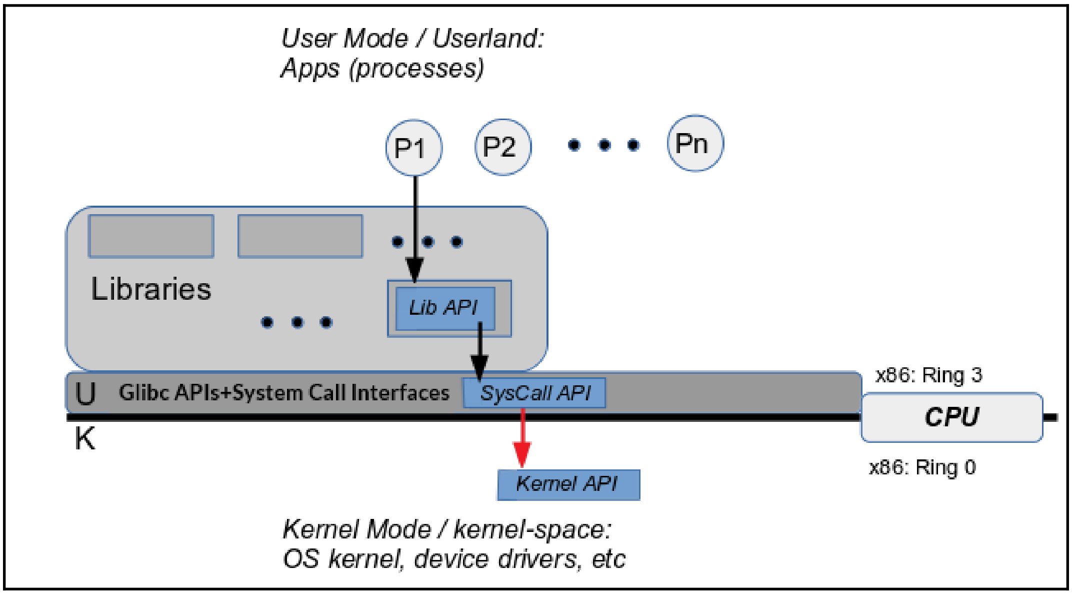
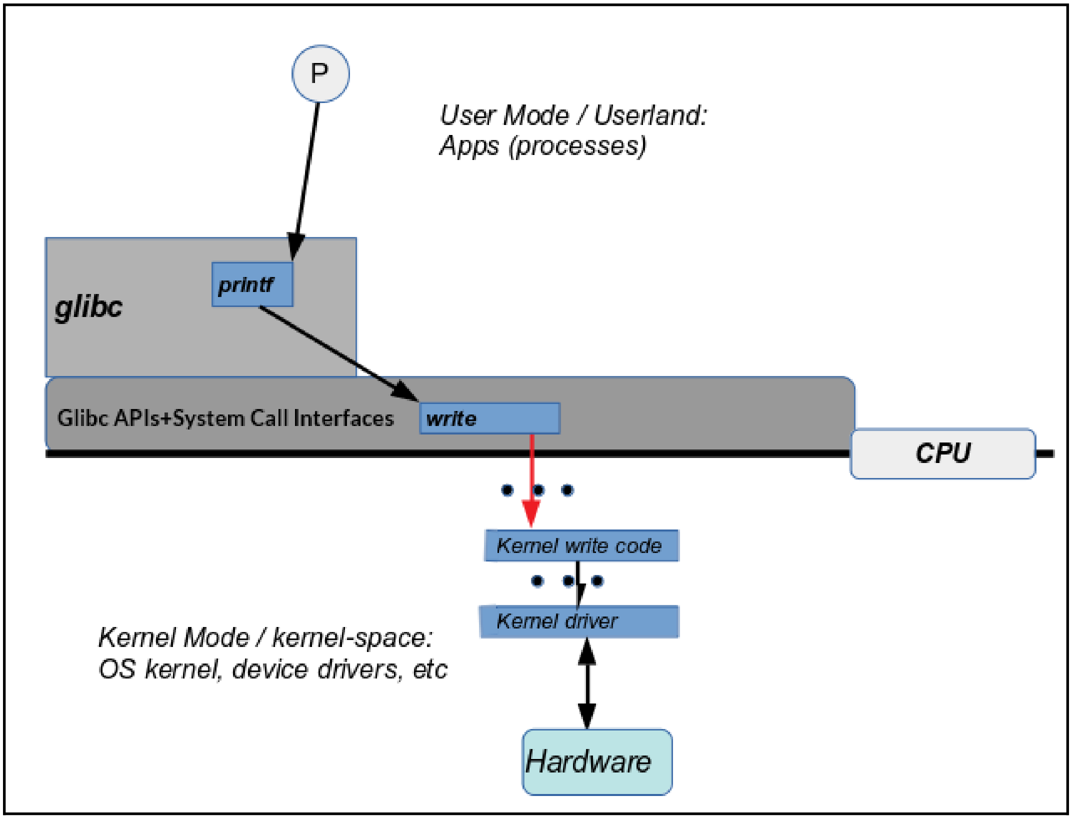
Chapter 1: Linux System Architecture

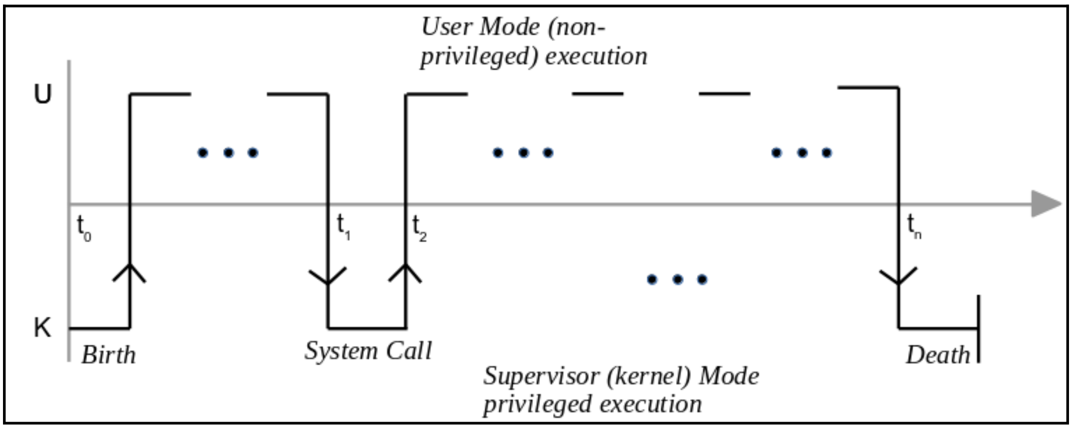




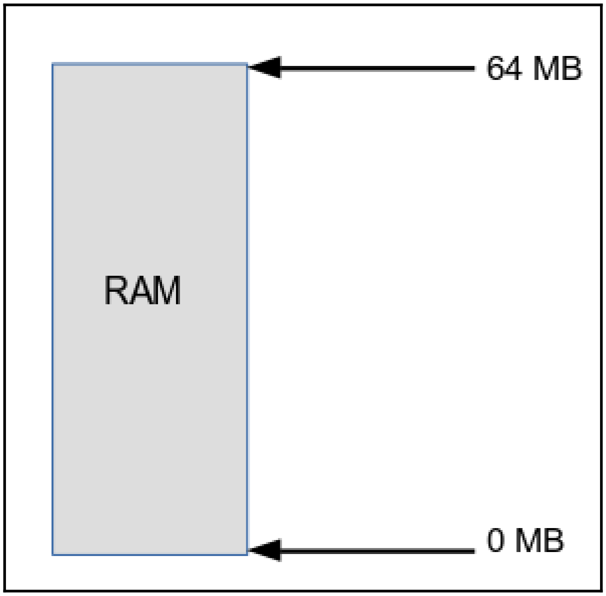

Application
Libraries
glibc / System Call Interface (SCI)
Operating System (OS) kernel, drivers, etc
Hardware Layer

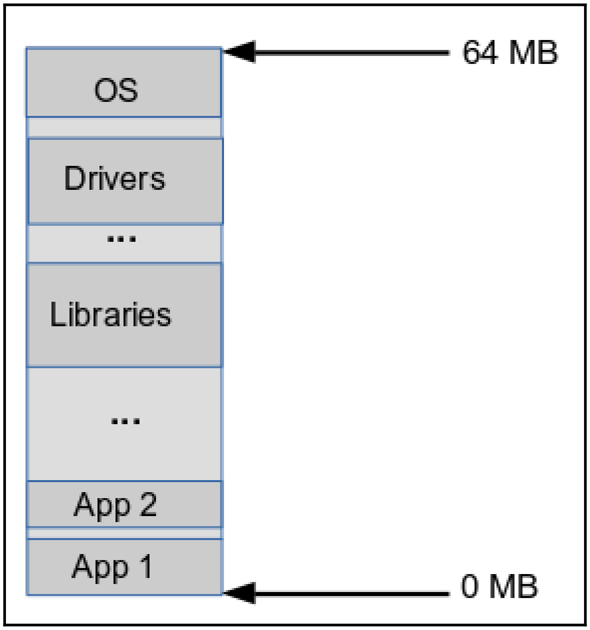

User Mode / Userland:
Apps (processes)
P1
P2
Pn
Libraries
Lib API
U
Glibc APIs+System Call Interfaces
SysCall API
x86: Ring 3
CPU
K
x86: Ring 0
Kernel API
Kernel Mode / kernel-space:
OS kernel, device drivers, etc

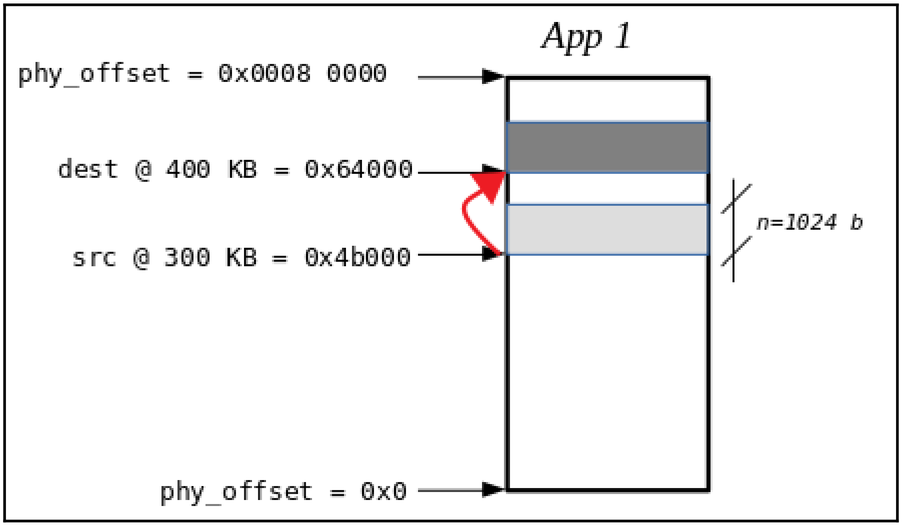

P
User Mode / Userland:
Apps (processes)
glibc
printf
Glibc APIs+System Call Interfaces
write
CPU
Kernel write code
Kernel driver
Kernel Mode / kernel-space:
OS kernel, device drivers, etc
Hardware

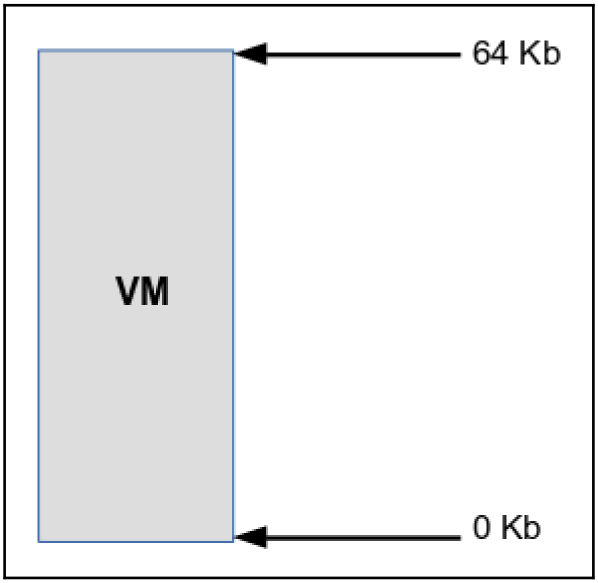

User Mode (non-
privileged) execution
U
K
t0
t1
t2
tn
Birth
System Call
Death
Supervisor (kernel) Mode
privileged execution

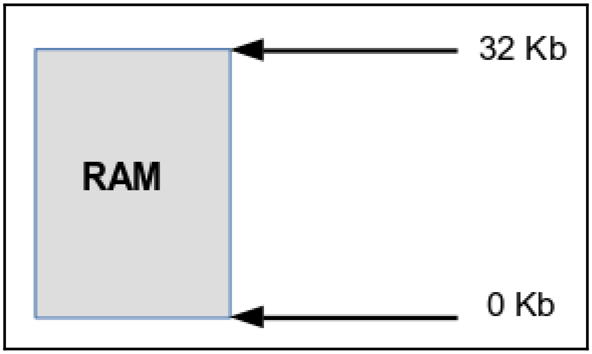
Chapter 2: Virtual Memory

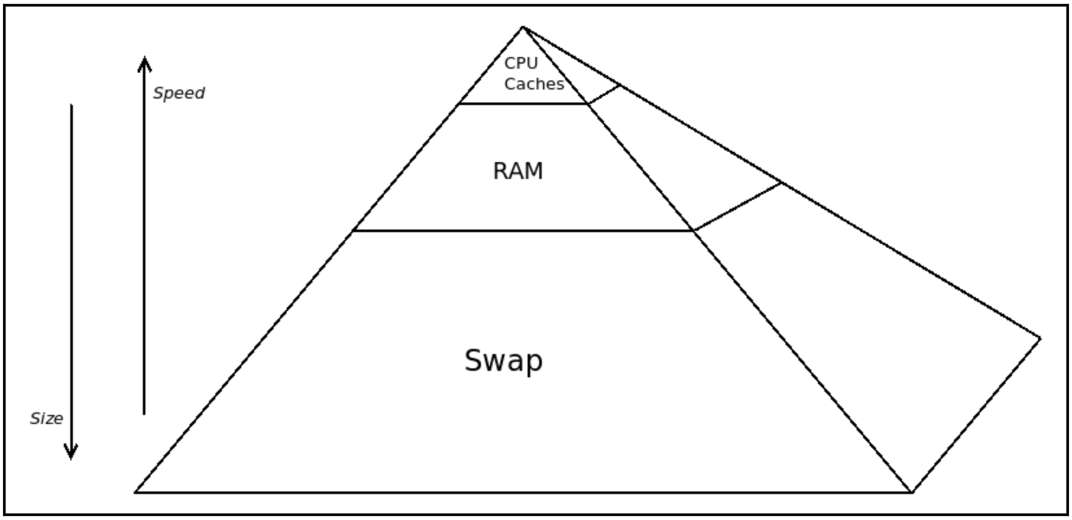


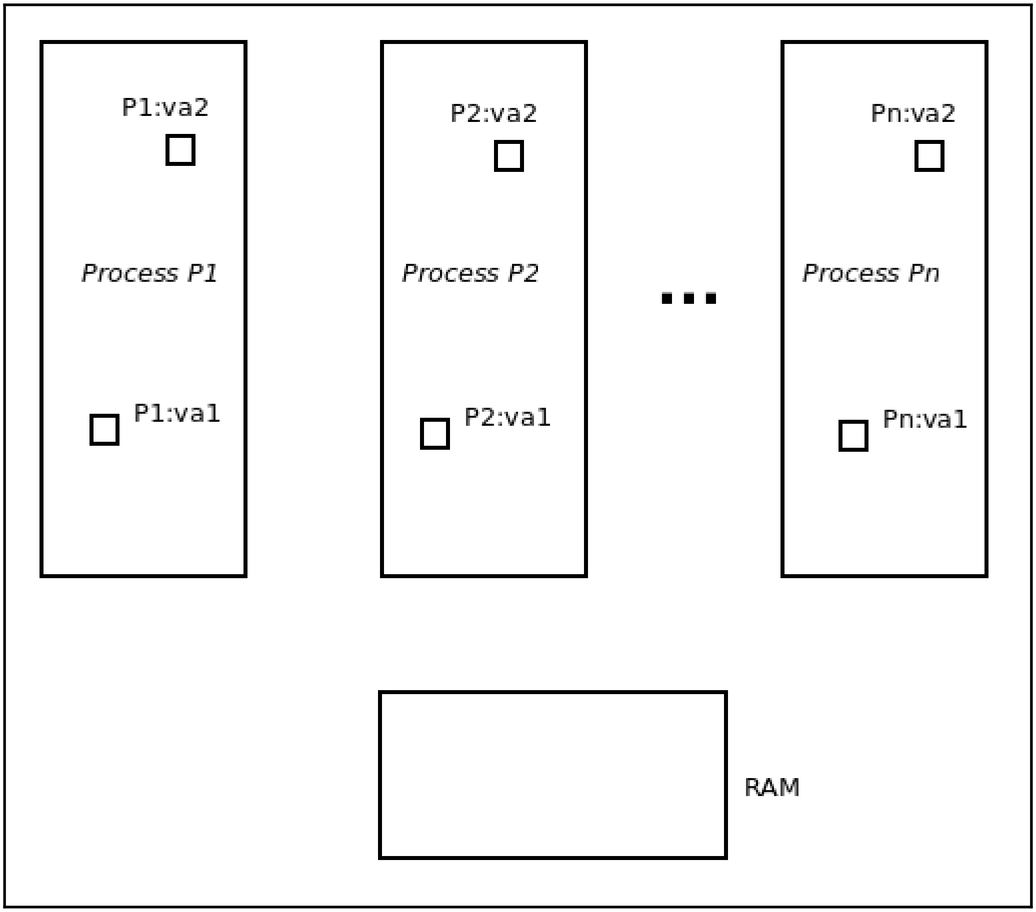


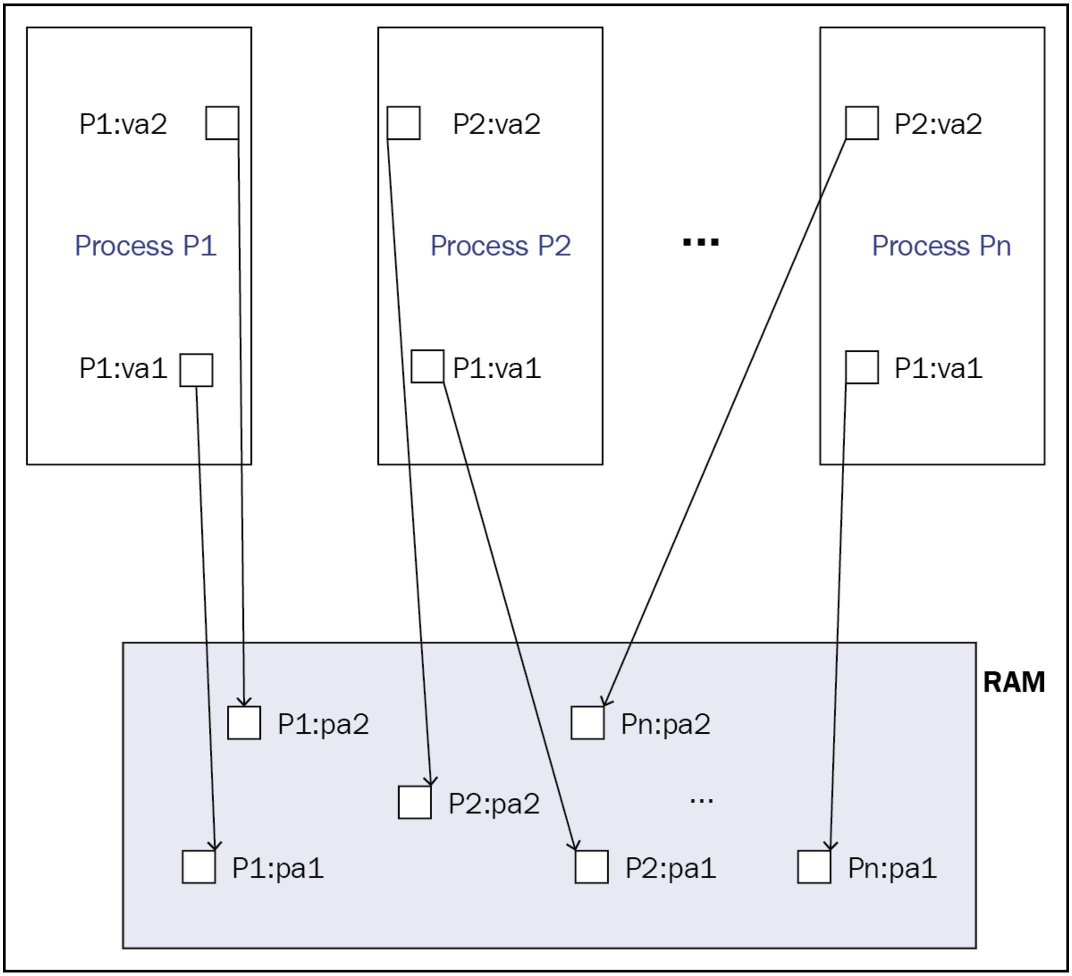

App 1
phy_offset = 0x0008 0000
dest @ 400 KB = 0x64000
src @ 300 KB = 0x4b000
n=1024 b
phy_offset = 0x0

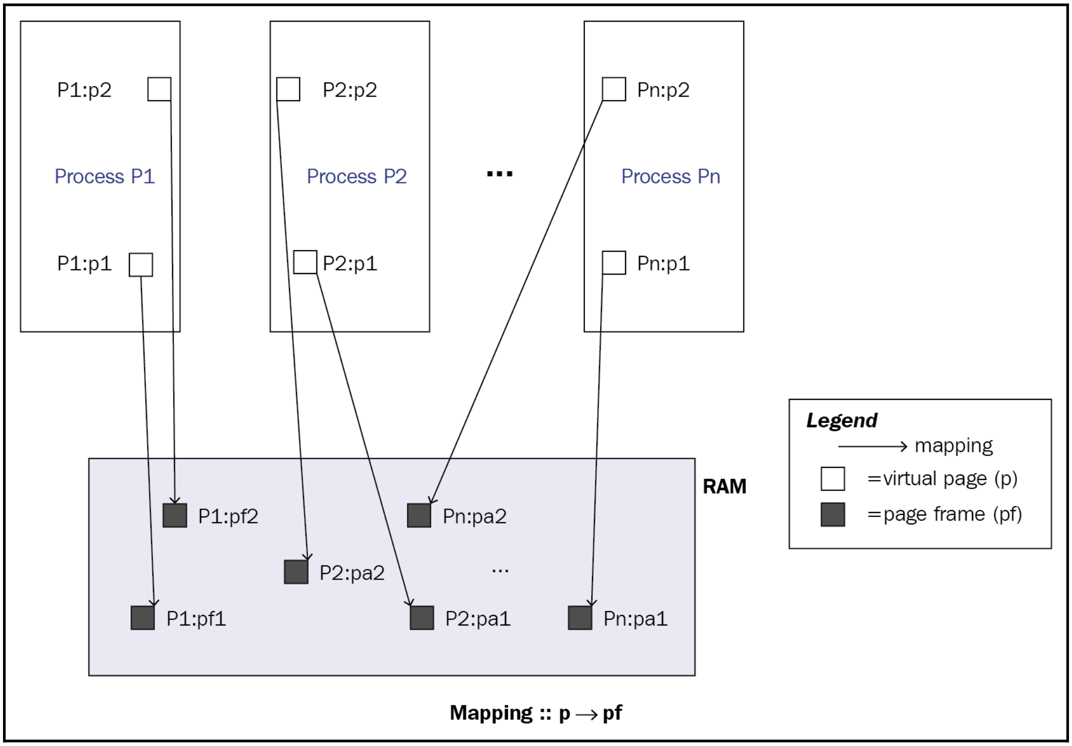

64 Kb
VM
0 Kb

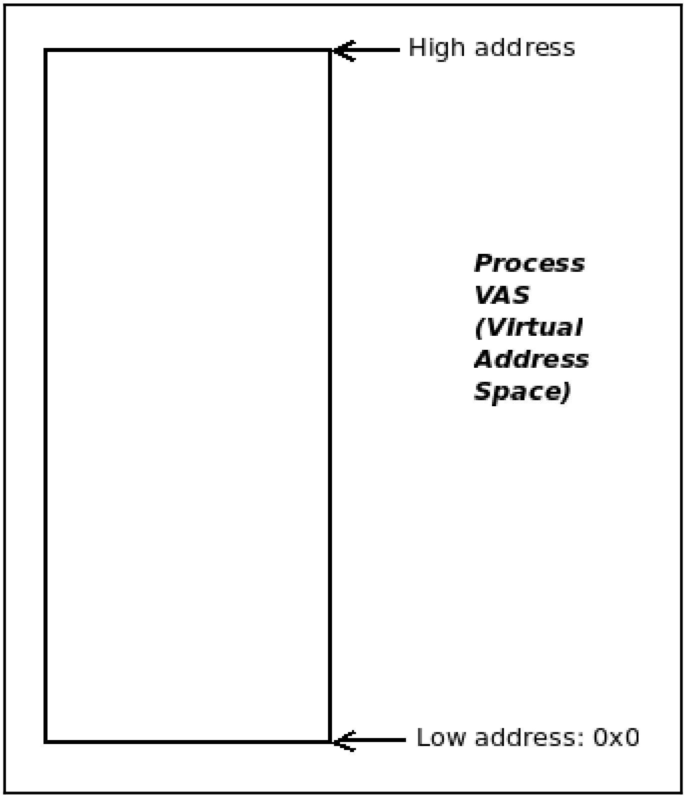

RAM
32 Kb
0 Kb

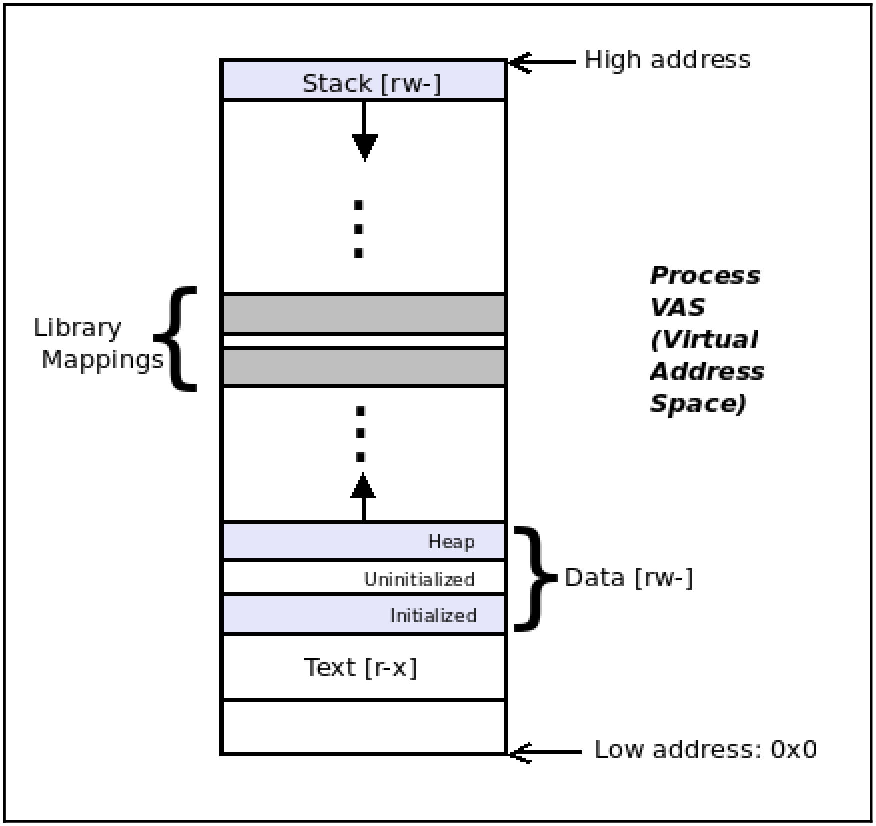

Speed
Size
CPU
Caches
RAM
Swap


P1:va2
Process P1
P1:va1
P2:va2
Process P2
P2:va1
...
Pn:va2
Process Pn
Pn:va1
RAM

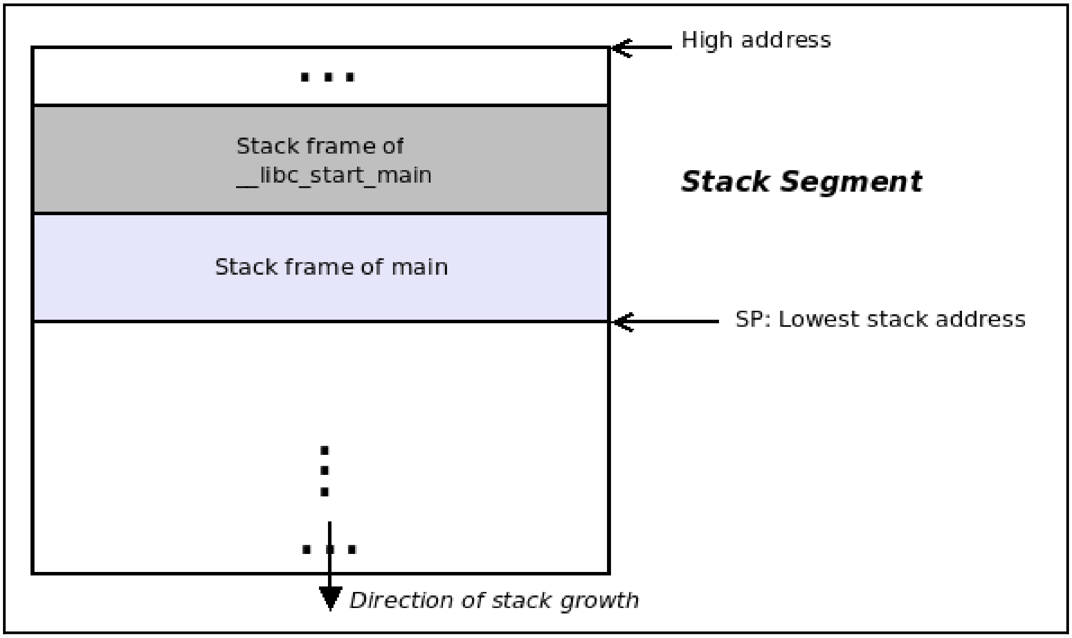

P1:va2
Process P1
P1:va1
P2:va2
Process P2
P1:va1
...
P2:va2
Process Pn
P1:va1
RAM
P1:pa2
Pn:pa2
P2:pa2
...
P1:pa1
P2:pa1
Pn:pa1

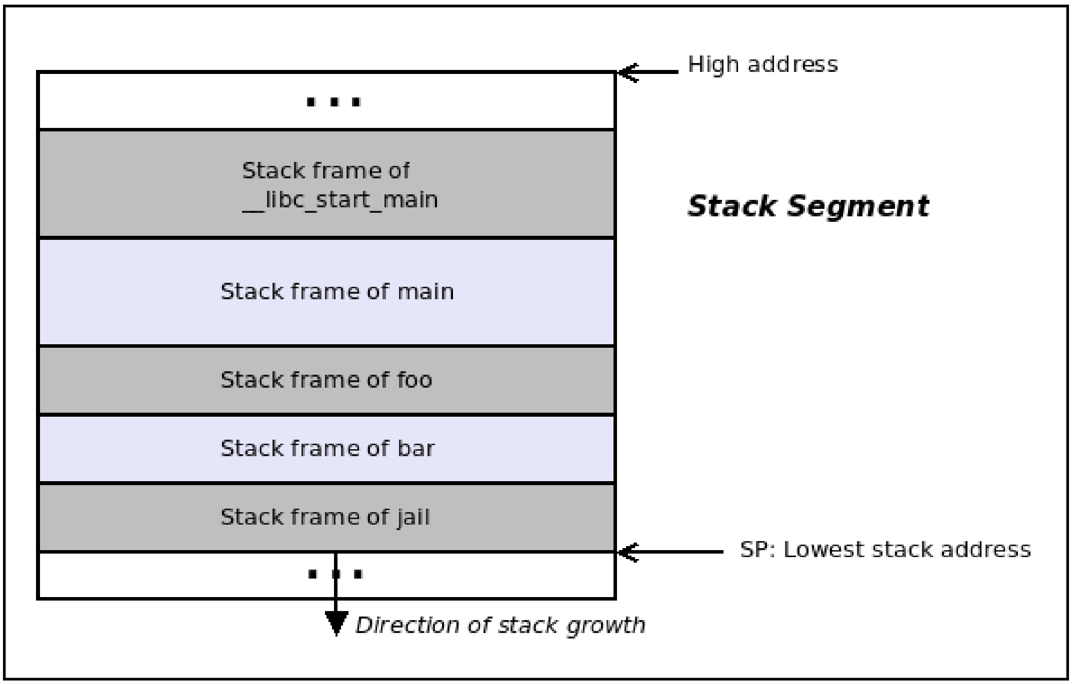

P1:p2
P2:p2
Pn:p2
Process P1
Process P2
...
Process Pn
P1:p1
P2:p1
Pn:p1
Legend
mapping
=virtual page (p)
=page frame (pf)
RAM
P1:pf2
Pn:pa2
P2:pa2
...
P1:pf1
P2:pa1
Pn:pa1
Mapping :: p → pf

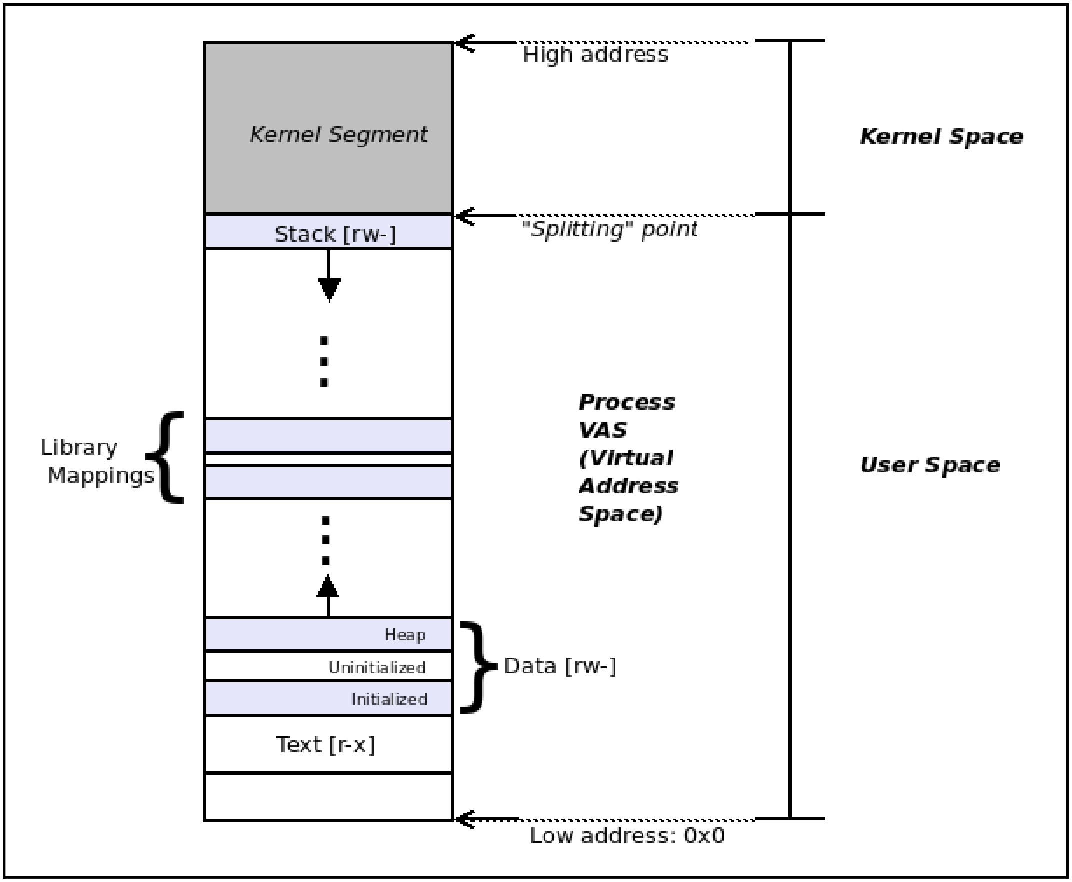

High address
Process
VAS
(Virtual
Address
Space)
Low address: 0x0

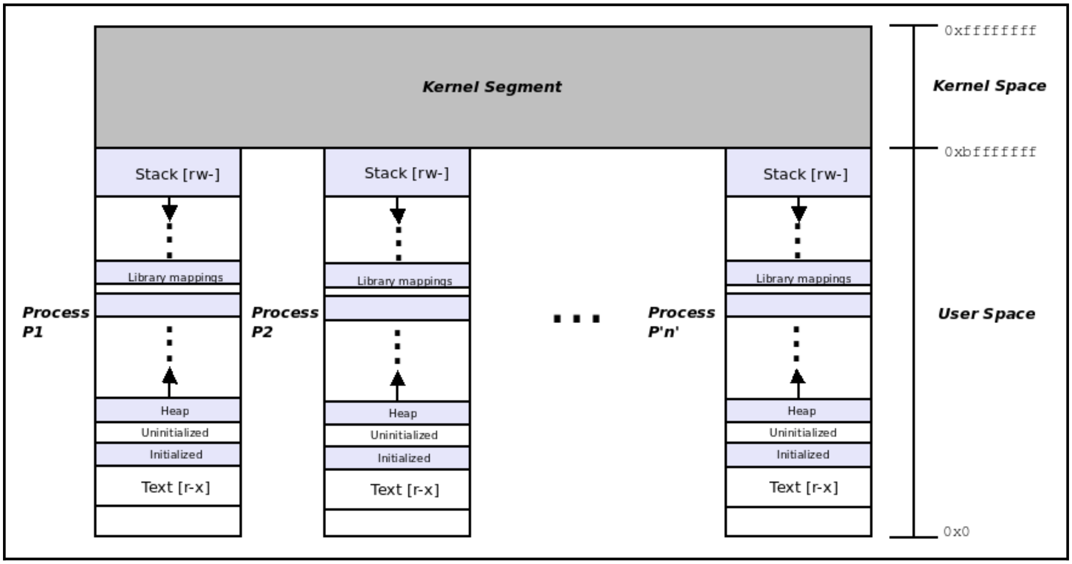


High address
Stack [rw-]
Process VAS (Virtual Address Space)
Library Mappings
Heap
Uninitialized
Initialized
Data [rw-]
Text [r-x]
Low address: 0x0


High address
...
Stack frame of
__libc_start_main
Stack Segment
Stack frame of main
SP: Lowest stack address
...
Direction of stack growth

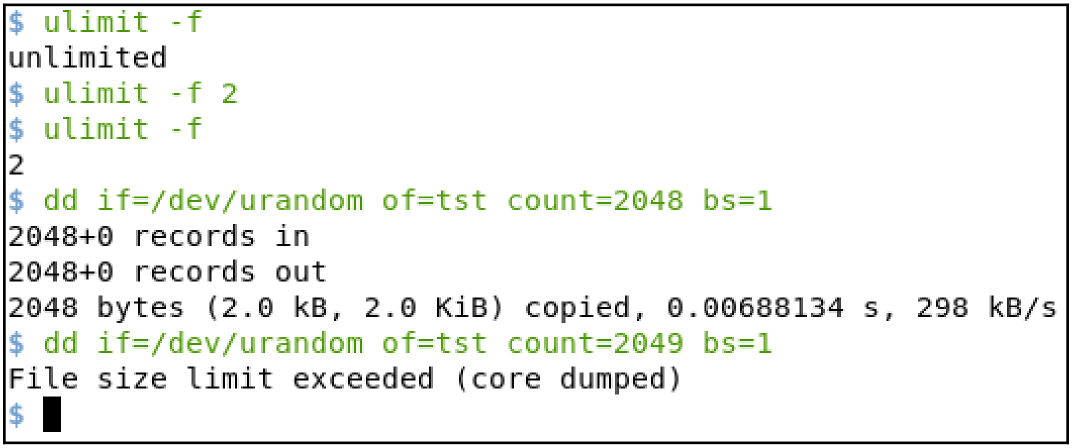

High address
...
Stack frame of
__libc_start_main
Stack Segment
Stack frame of main
Stack frame of foo
Stack frame of bar
Stack frame of jail
SP: Lowest stack address
...
Direction of stack growth


High address
Kernel Segment
Kernel Space
Stack [rw-]
"Splitting" point
Library
Mappings
Process
VAS
(Virtual
Address
Space)
User Space
Heap
Uninitialized
Initialized
Data [rw-]
Text [r-x]
Low address: 0x0

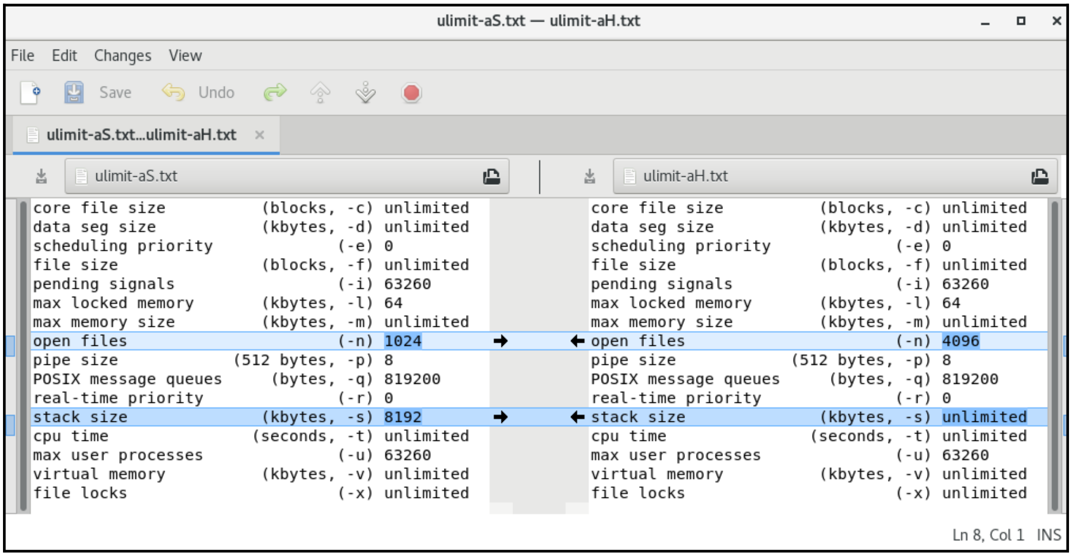


Kernel Segment
0xffffffff
Kernel Space
0xbfffffff
Stack [rw-]
Library mappings
Process P1
Heap
Uninitialized
Initialized
Text [r-x]
Stack [rw-]
Library mappings
Process P2
Heap
Uninitialized
Initialized
Text [r-x]
Process P'n'
Stack [rw-]
Library mappings
Heap
Uninitialized
Initialized
Text [r-x]
User Space
0x0

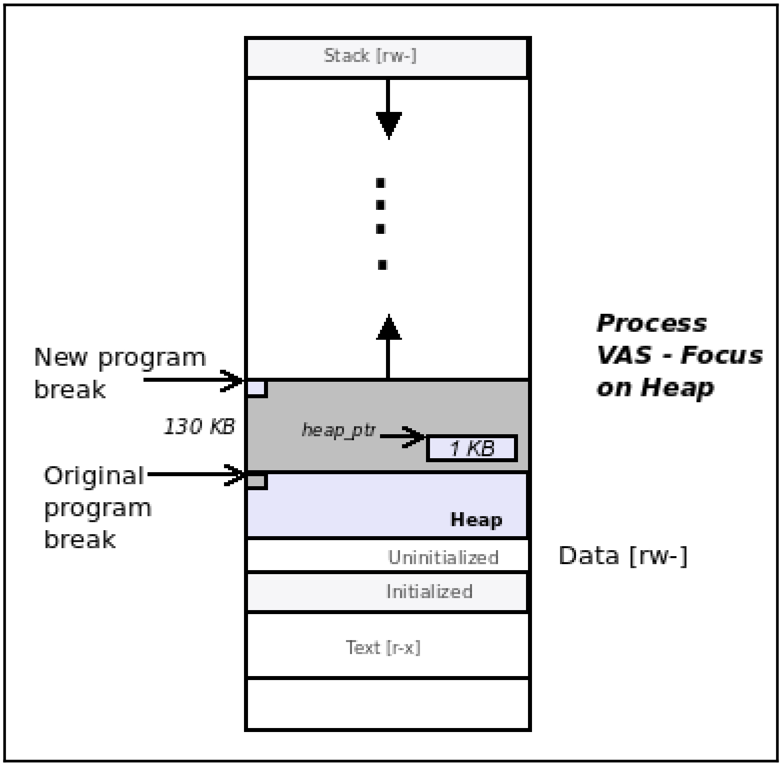
Chapter 3: Resource Limits



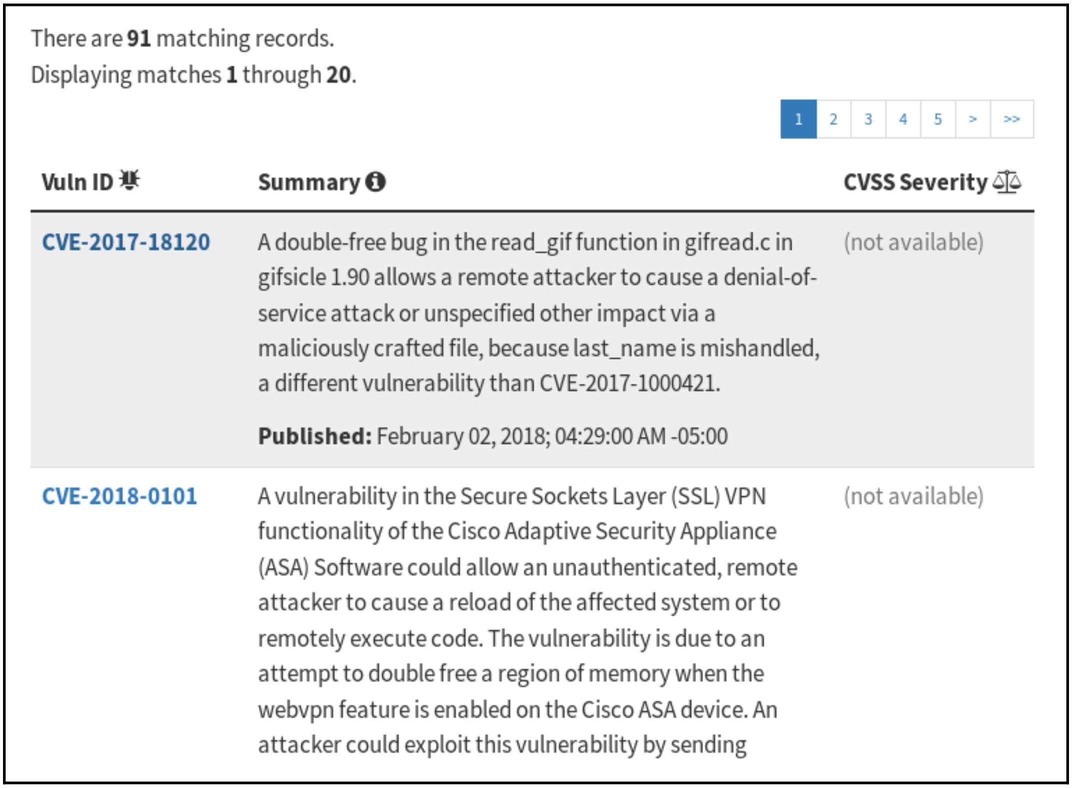
```
$ ulimit -f
unlimited
$ ulimit -f 2
$ ulimit -f
2
$ dd if=/dev/urandom of=tst count=2048 bs=1
2048+0 records in
2048+0 records out
2048 bytes (2.0 kB, 2.0 KiB) copied, 0.00688134 s, 298 kB/s
$ dd if=/dev/urandom of=tst count=2049 bs=1
File size limit exceeded (core dumped)
$
```



```
RESOURCE   DESCRIPTION                             SOFT      HARD UNITS
AS         address space limit                unlimited unlimited bytes
CORE       max core file size                         0 unlimited blocks
CPU        CPU time                           unlimited unlimited seconds
DATA       max data size                      unlimited unlimited bytes
FSIZE      max file size                           2048      2048 blocks
LOCKS      max number of file locks held      unlimited unlimited
MEMLOCK    max locked-in-memory address space     65536     65536 bytes
MSGQUEUE   max bytes in POSIX mqueues            819200    819200 bytes
NICE       max nice prio allowed to raise             0         0
NOFILE     max number of open files                2000      2000
NPROC      max number of processes                 7741      7741
RSS        max resident set size              unlimited unlimited pages
RTPRIO     max real-time priority                     0         0
RTTIME     timeout for real-time tasks        unlimited unlimited microsecs
SIGPENDING max number of pending signals           7741      7741
STACK      max stack size                       8388608 unlimited bytes
```




```
22875  3) memprot-29591  |  d...+ 10.204 us  |  }
22876  3) memprot-29591  |  d...             |  do_syscall_64() {
22877  3) memprot-29591  |  ....             |    SyS_mprotect() {
22878  3) memprot-29591  |  ....             |      do_mprotect_pkey() {
22879  3) memprot-29591  |  ....             |        down_write_killable() {
22880  3) memprot-29591  |  ....             |          _cond_resched() {
22881  3) memprot-29591  |  ....  0.034 us   |            rcu_all_qs();
22882  3) memprot-29591  |  ....  0.321 us   |          }
22883  3) memprot-29591  |  ....  0.616 us   |        }
22884  3) memprot-29591  |  ....             |        find_vma() {
22885  3) memprot-29591  |  ....  0.107 us   |          vmacache_find();
22886  3) memprot-29591  |  ....  0.116 us   |          vmacache_update();
22887  3) memprot-29591  |  ....  1.227 us   |        }
22888  3) memprot-29591  |  ....             |        security_file_mprotect() {
22889  3) memprot-29591  |  ....             |          selinux_file_mprotect() {
22890  3) memprot-29591  |  ....             |            avc_has_perm() {
22891  3) memprot-29591  |  ....  0.047 us   |              avc_denied();
22892  3) memprot-29591  |  ....             |              slow_avc_audit() {
22893  3) memprot-29591  |  ....             |                common_lsm_audit() {
```



There are **91** matching records.
Displaying matches **1** through **20**.

1 2 3 4 5 > >>

Vuln ID	Summary	CVSS Severity
CVE-2017-18120	A double-free bug in the read_gif function in gifread.c in gifsicle 1.90 allows a remote attacker to cause a denial-of-service attack or unspecified other impact via a maliciously crafted file, because last_name is mishandled, a different vulnerability than CVE-2017-1000421. **Published:** February 02, 2018; 04:29:00 AM -05:00	(not available)
CVE-2018-0101	A vulnerability in the Secure Sockets Layer (SSL) VPN functionality of the Cisco Adaptive Security Appliance (ASA) Software could allow an unauthenticated, remote attacker to cause a reload of the affected system or to remotely execute code. The vulnerability is due to an attempt to double free a region of memory when the webvpn feature is enabled on the Cisco ASA device. An attacker could exploit this vulnerability by sending	(not available)

```
$ valgrind ./membugs_dbg 8
==24337== Memcheck, a memory error detector
==24337== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==24337== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==24337== Command: ./membugs_dbg 8
==24337==
uaf():165: arr = 0x521f040:aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
==24337== Invalid write of size 1
==24337==    at 0x4C32E7F: strncpy (vg_replace_strmem.c:549)
==24337==    by 0x400FB5: uaf (membugs.c:173)
==24337==    by 0x4014A6: process_args (membugs.c:348)
==24337==    by 0x401574: main (membugs.c:379)
==24337==  Address 0x521f040 is 0 bytes inside a block of size 512 free'd
==24337==    at 0x4C30D18: free (vg_replace_malloc.c:530)
==24337==    by 0x400F93: uaf (membugs.c:171)
==24337==    by 0x4014A6: process_args (membugs.c:348)
==24337==    by 0x401574: main (membugs.c:379)
==24337==  Block was alloc'd at
==24337==    at 0x4C2FB6B: malloc (vg_replace_malloc.c:299)
==24337==    by 0x400EC5: uaf (membugs.c:159)
==24337==    by 0x4014A6: process_args (membugs.c:348)
==24337==    by 0x401574: main (membugs.c:379)
==24337==
==24337== Invalid read of size 1
==24337==    at 0x4C32B2B: strnlen (vg_replace_strmem.c:425)
==24337==    by 0x4E8FFB2: vfprintf (in /usr/lib64/libc-2.26.so)
==24337==    by 0x4E99255: printf (in /usr/lib64/libc-2.26.so)
==24337==    by 0x400FE2: uaf (membugs.c:174)
==24337==    by 0x4014A6: process_args (membugs.c:348)
==24337==    by 0x401574: main (membugs.c:379)
```

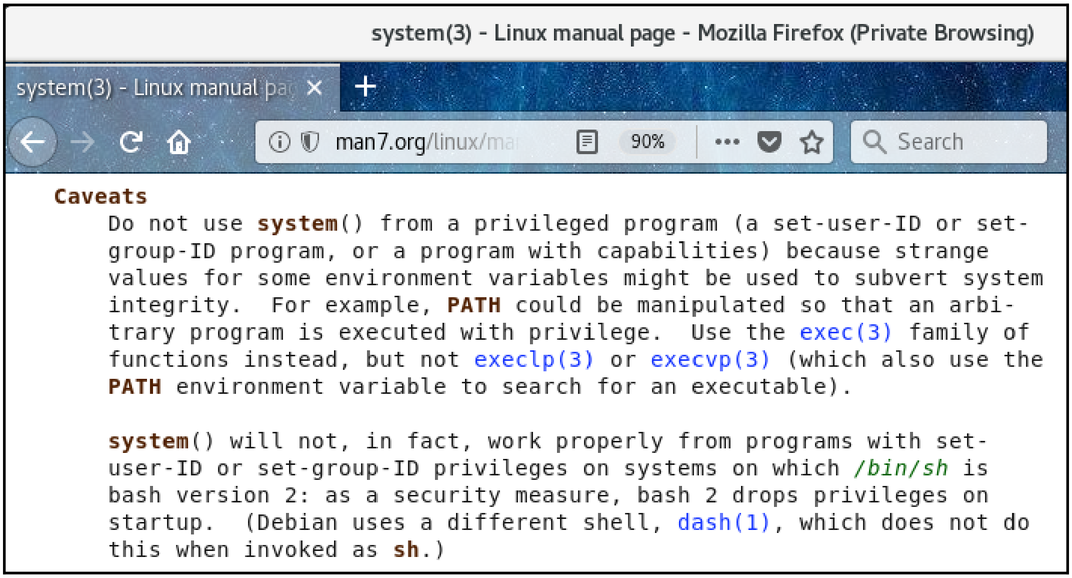
```
$ ./membugs_dbg_asan 2
=================================================================
==4125==ERROR: AddressSanitizer: stack-buffer-overflow on address 0x7ffe9f8d8174 at pc 0x00000051271d bp 0x7ffe9f8d8130 sp 0x7
ffe9f8d8128
WRITE of size 4 at 0x7ffe9f8d8174 thread T0
    #0 0x51271c  (/home/seawolf/0tmp/membugs_dbg_asan+0x51271c)
    #1 0x51244e  (/home/seawolf/0tmp/membugs_dbg_asan+0x51244e)
    #2 0x512291  (/home/seawolf/0tmp/membugs_dbg_asan+0x512291)
    #3 0x7fe3408ccb96  (/lib/x86_64-linux-gnu/libc.so.6+0x21b96)
    #4 0x419ea9  (/home/seawolf/0tmp/membugs_dbg_asan+0x419ea9)

Address 0x7ffe9f8d8174 is located in stack of thread T0 at offset 52 in frame
    #0 0x5125ef  (/home/seawolf/0tmp/membugs_dbg_asan+0x5125ef)

  This frame has 1 object(s):
    [32, 52) 'arr' (line 265) <== Memory access at offset 52 overflows this variable
HINT: this may be a false positive if your program uses some custom stack unwind mechanism or swapcontext
      (longjmp and C++ exceptions *are* supported)
SUMMARY: AddressSanitizer: stack-buffer-overflow (/home/seawolf/0tmp/membugs_dbg_asan+0x51271c)
Shadow bytes around the buggy address:
  0x100053f12fd0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100053f12fe0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100053f12ff0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100053f13000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100053f13010: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
=>0x100053f13020: 00 00 00 00 00 00 00 00 f1 f1 f1 f1 00 00[04]f3
  0x100053f13030: f3 f3 f3 f3 00 00 00 00 00 00 00 00 00 00 00 00
  0x100053f13040: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100053f13050: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100053f13060: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
  0x100053f13070: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
Shadow byte legend (one shadow byte represents 8 application bytes):
  Addressable:           00
```

```
$ ./membugs_dbg_asan 13

## Leakage test: case 3: "lib" API: runtime cond = 0
mypath = /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local
/games:/snap/bin:/sbin:/usr/sbin:/usr/local/sbin:/home/seawolf/kaiwanTECH/usefulsnips:/usr/
lib/llvm-6.0/bin/

## Leakage test: case 3: "lib" API: runtime cond = 1
mypath = /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local
/games:/snap/bin:/sbin:/usr/sbin:/usr/local/sbin:/home/seawolf/kaiwanTECH/usefulsnips:/usr/
lib/llvm-6.0/bin/

=================================================================
==26220==ERROR: LeakSanitizer: detected memory leaks

Direct leak of 4096 byte(s) in 1 object(s) allocated from:
    #0 0x4d9d60 in malloc (/home/seawolf/0tmp/membugs_dbg_asan+0x4d9d60)
    #1 0x513eca in silly_getpath /home/seawolf/kaiwanTECH/book_src/ch5/membugs.c:37:9
    #2 0x513d0d in leakage_case3 /home/seawolf/kaiwanTECH/book_src/ch5/membugs.c:55:2
    #3 0x512522 in process_args /home/seawolf/kaiwanTECH/book_src/ch5/membugs.c:363:4
    #4 0x512291 in main /home/seawolf/kaiwanTECH/book_src/ch5/membugs.c:375:2
    #5 0x7f5f23247b96 in __libc_start_main /build/glibc-OTsEL5/glibc-2.27/csu/../csu/libc-s
tart.c:310

SUMMARY: AddressSanitizer: 4096 byte(s) leaked in 1 allocation(s).
$
```

Chapter 7: Process Credentials

Access Category	Owner (U)			Group (G)			Others (O)		
Permission Bits	r	w	x	r	w	x	r	w	x
Example (myfile)	1	1	0	1	1	0	1	0	0

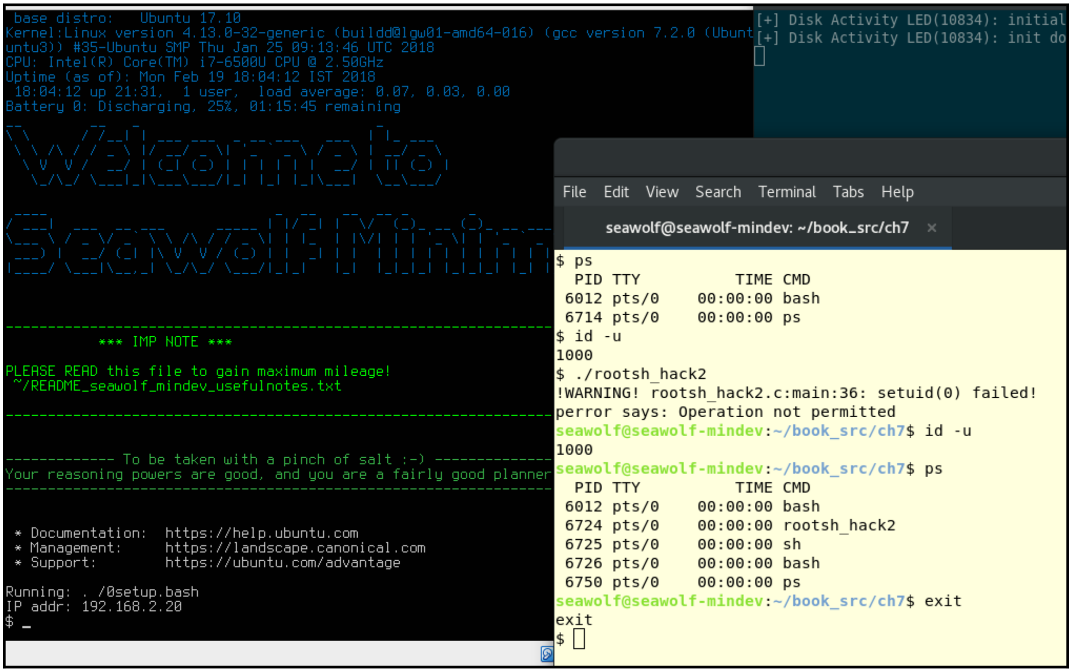
```
$ ls -l $(which passwd)
-rwsr-xr-x 1 root root 54224 Aug 21 05:26 /usr/bin/passwd*
$ 
```

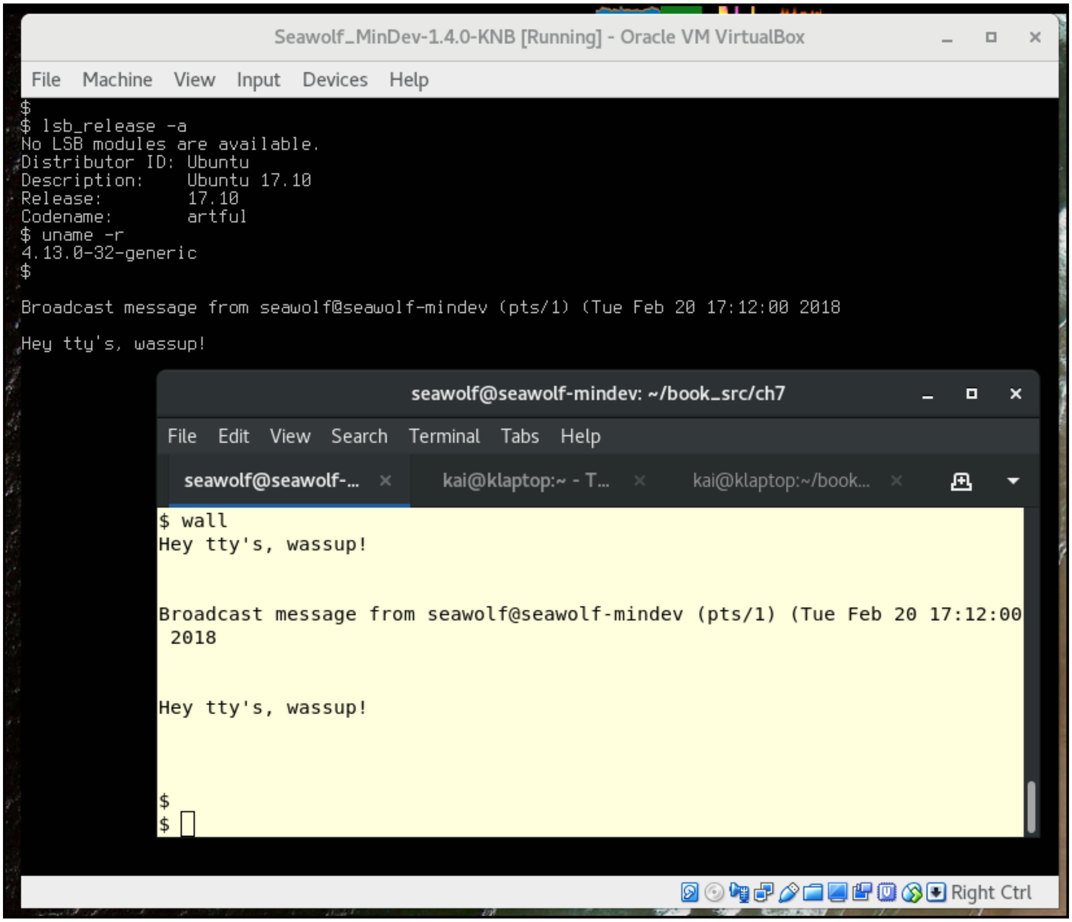


Caveats

Do not use **system**() from a privileged program (a set-user-ID or set-group-ID program, or a program with capabilities) because strange values for some environment variables might be used to subvert system integrity. For example, **PATH** could be manipulated so that an arbitrary program is executed with privilege. Use the exec(3) family of functions instead, but not execlp(3) or execvp(3) (which also use the **PATH** environment variable to search for an executable).

system() will not, in fact, work properly from programs with set-user-ID or set-group-ID privileges on systems on which */bin/sh* is bash version 2: as a security measure, bash 2 drops privileges on startup. (Debian uses a different shell, dash(1), which does not do this when invoked as **sh**.)


base distro: Ubuntu 17.10
Kernel:Linux version 4.13.0-32-generic (buildd@lgw01-amd64-016) (gcc version 7.2.0 (Ubuntu untu3)) #35-Ubuntu SMP Thu Jan 25 09:13:46 UTC 2018
CPU: Intel(R) Core(TM) i7-6500U CPU @ 2.50GHz
Uptime (as of): Mon Feb 19 18:04:12 IST 2018
18:04:12 up 21:31, 1 user, load average: 0.07, 0.03, 0.00
Battery 0: Discharging, 25%, 01:15:45 remaining
*** IMP NOTE ***
PLEASE READ this file to gain maximum mileage!
~/README_seawolf_mindev_usefulnotes.txt
------------ To be taken with a pinch of salt :-) ------------
Your reasoning powers are good, and you are a fairly good planner
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/advantage
Running: . /0setup.bash
IP addr: 192.168.2.20
$ _
[+] Disk Activity LED(10834): initial
[+] Disk Activity LED(10834): init do
File Edit View Search Terminal Tabs Help
seawolf@seawolf-mindev: ~/book_src/ch7
$ ps
PID TTY TIME CMD
6012 pts/0 00:00:00 bash
6714 pts/0 00:00:00 ps
$ id -u
1000
$ ./rootsh_hack2
!WARNING! rootsh_hack2.c:main:36: setuid(0) failed!
perror says: Operation not permitted
seawolf@seawolf-mindev:~/book_src/ch7$ id -u
1000
seawolf@seawolf-mindev:~/book_src/ch7$ ps
PID TTY TIME CMD
6012 pts/0 00:00:00 bash
6724 pts/0 00:00:00 rootsh_hack2
6725 pts/0 00:00:00 sh
6726 pts/0 00:00:00 bash
6750 pts/0 00:00:00 ps
seawolf@seawolf-mindev:~/book_src/ch7$ exit
exit
$


Seawolf_MinDev-1.4.0-KNB [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
$
$ lsb_release -a
No LSB modules are available.
Distributor ID: Ubuntu
Description: Ubuntu 17.10
Release: 17.10
Codename: artful
$ uname -r
4.13.0-32-generic
$
Broadcast message from seawolf@seawolf-mindev (pts/1) (Tue Feb 20 17:12:00 2018
Hey tty's, wassup!
seawolf@seawolf-mindev: ~/book_src/ch7
File Edit View Search Terminal Tabs Help
seawolf@seawolf-...
kai@klaptop:~ - T...
kai@klaptop:~/book...
$ wall
Hey tty's, wassup!
Broadcast message from seawolf@seawolf-mindev (pts/1) (Tue Feb 20 17:12:00 2018
Hey tty's, wassup!
$
$
Right Ctrl

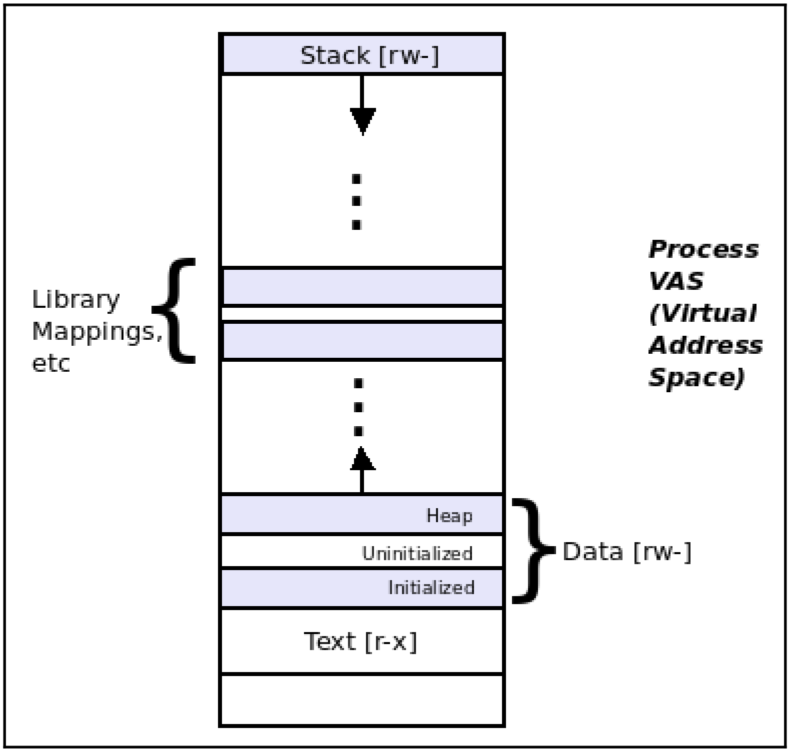
```
$ sudo chown root savedset_demo
[sudo] password for seawolf:
$ sudo chmod u+s savedset_demo
$ ls -l savedset_demo
-rwsrwxr-x 1 root seawolf 13144 Feb 20 09:22 savedset_demo*
$ ./savedset_demo
t0: Init:
RUID=1000 EUID=0
RGID=1000 EGID=1000
 Ok, we're effectively running as root! (EUID==0)
t1: Becoming my original self!
RUID=1000 EUID=1000
RGID=1000 EGID=1000
t2: Switching to privileged state now...
RUID=1000 EUID=0
RGID=1000 EGID=1000
 Yup, we're root again!
t3: Switching back to unprivileged state now ...
RUID=1000 EUID=1000
RGID=1000 EGID=1000
$ 
```

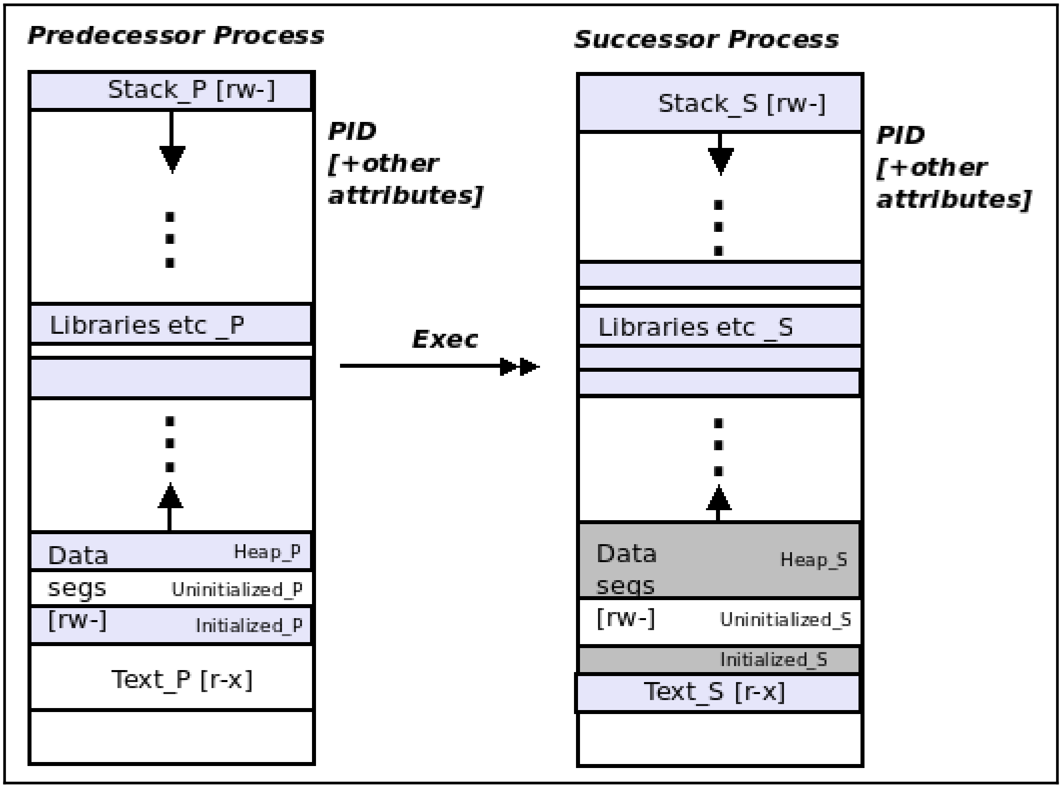
Chapter 8: Process Capabilities

```
Transformation of capabilities during execve()
    During an execve(2), the kernel calculates the new capabilities of the  process
    using the following algorithm:

        P'(ambient)     = (file is privileged) ? 0 : P(ambient)

        P'(permitted)   = (P(inheritable) & F(inheritable)) |
                          (F(permitted) & cap_bset) | P'(ambient)

        P'(effective)   = F(effective) ? P'(permitted) : P'(ambient)

        P'(inheritable) = P(inheritable)    [i.e., unchanged]

    where:

        P         denotes the value of a thread capability set before the execve(2)

        P'        denotes the value of a thread capability set after the execve(2)

        F         denotes a file capability set

        cap_bset  is the value of the capability bounding set (described below).
```

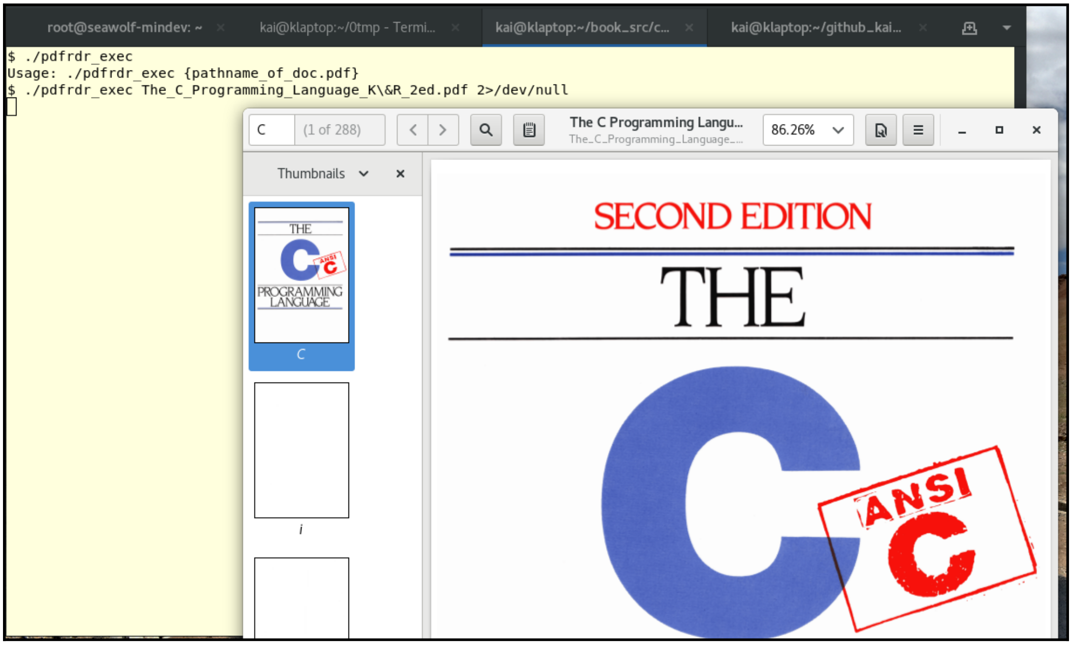
```
$ ls -l /usr/bin/passwd /usr/bin/write /usr/bin/ping /bin/dumpcap
-rwxr-x---. 1 root wireshark 109344 Jan 19 19:45 /bin/dumpcap
-rwsr-xr-x. 1 root root       27880 Aug  4  2017 /usr/bin/passwd
-rwxr-xr-x. 1 root root       62080 Aug  3  2017 /usr/bin/ping
-rwxr-sr-x. 1 root tty        19584 Sep 22 14:07 /usr/bin/write
$
```

Chapter 9: Process Execution

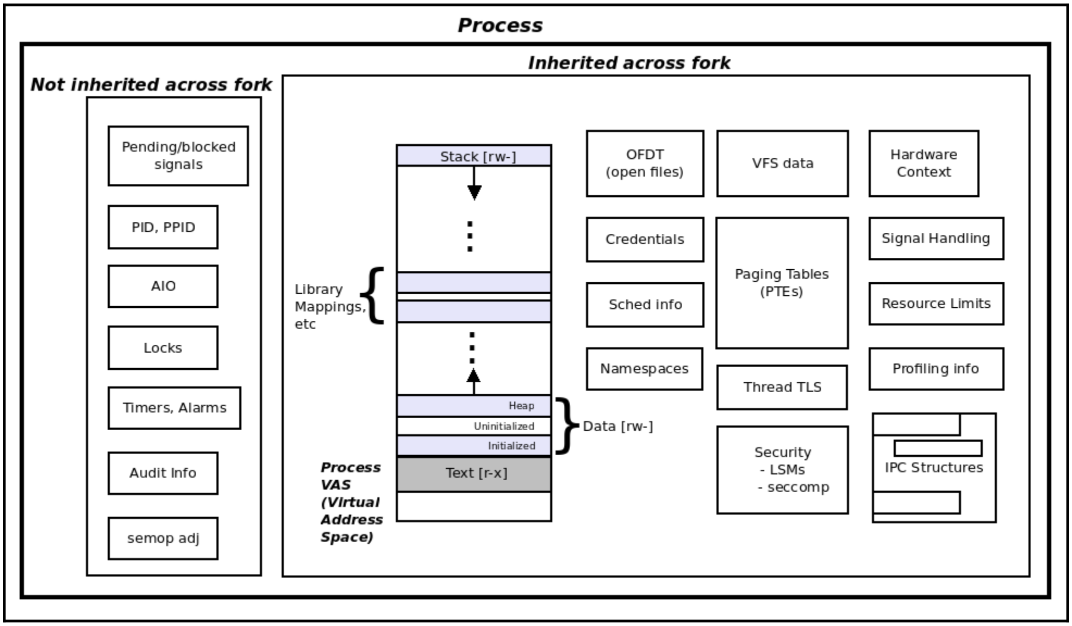


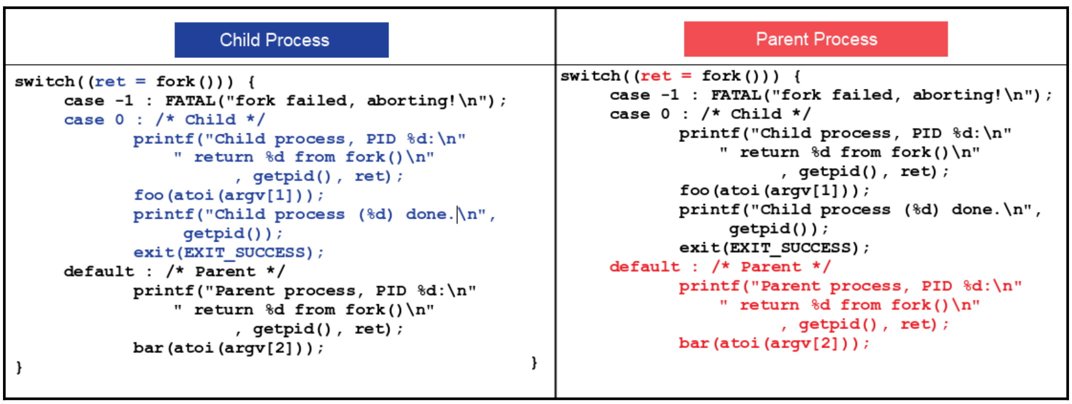



```
$ ps
  PID TTY          TIME CMD
 3396 pts/3    00:00:00 bash
13030 pts/3    00:00:00 ps
$ bash
$ ps
  PID TTY          TIME CMD
 3396 pts/3    00:00:00 bash
13040 pts/3    00:00:00 bash
13087 pts/3    00:00:00 ps
$ exec ps
  PID TTY          TIME CMD
 3396 pts/3    00:00:00 bash
13040 pts/3    00:00:00 ps
$ █
```

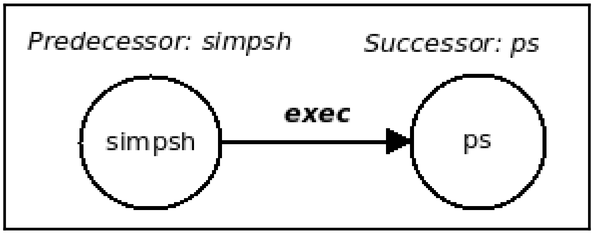

root@seawolf-mindev: ~
kai@klaptop:~/0tmp - Termi...
kai@klaptop:~/book_src/c...
kai@klaptop:~/github_kai...
$ ./pdfrdr_exec
Usage: ./pdfrdr_exec {pathname_of_doc.pdf}
$ ./pdfrdr_exec The_C_Programming_Language_K\&R_2ed.pdf 2>/dev/null
C
(1 of 288)
The C Programming Langu...
The_C_Programming_Language_...
86.26%
Thumbnails
THE
C
ANSI
C
PROGRAMMING
LANGUAGE
C
i
SECOND EDITION
THE
C
ANSI
C

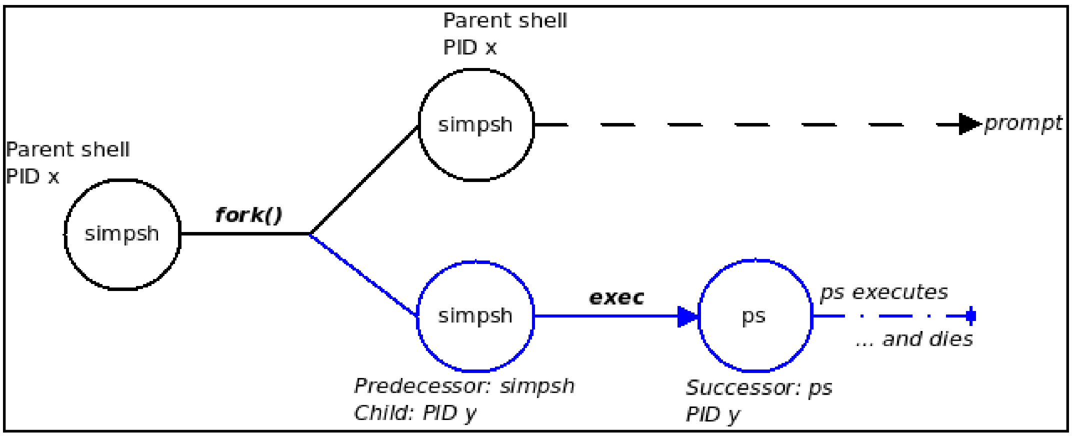
Chapter 10: Process Creation

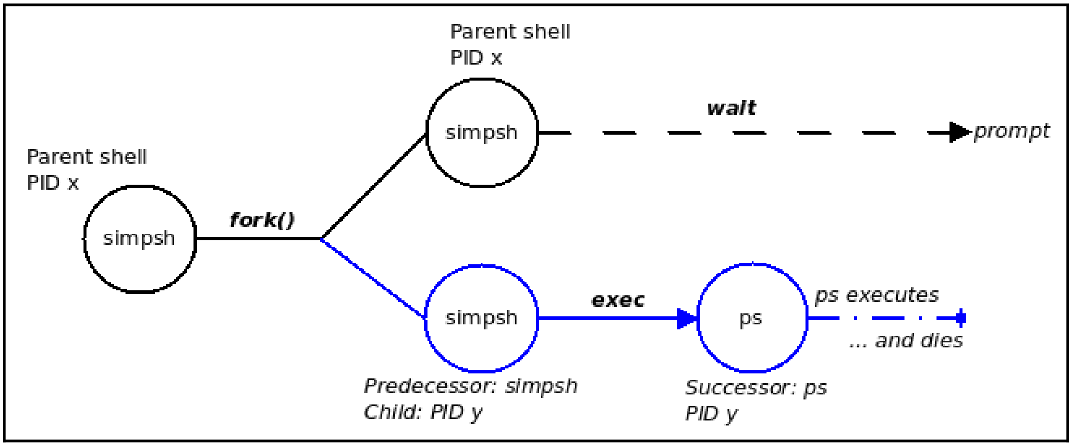




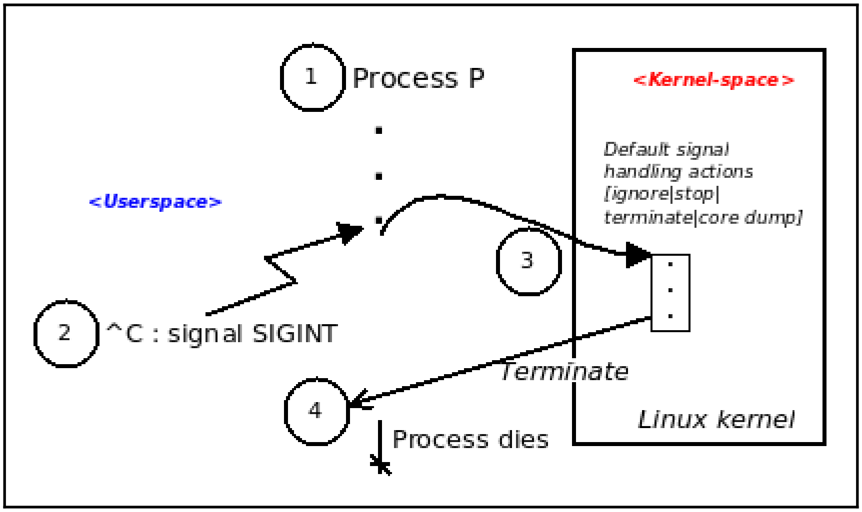
```
switch((ret = fork())) {
     case -1 : FATAL("fork failed, aborting!\n");
     case 0 : /* Child */
            printf("Child process, PID %d:\n"
                " return %d from fork()\n"
                      , getpid(), ret);
            foo(atoi(argv[1]));
            printf("Child process (%d) done, exiting ...\n",
                 getpid());
            exit(EXIT_SUCCESS);
     default : /* Parent */
            printf("Parent process, PID %d:\n"
                " return %d from fork()\n"
                      , getpid(), ret);
            bar(atoi(argv[2]));
}
```

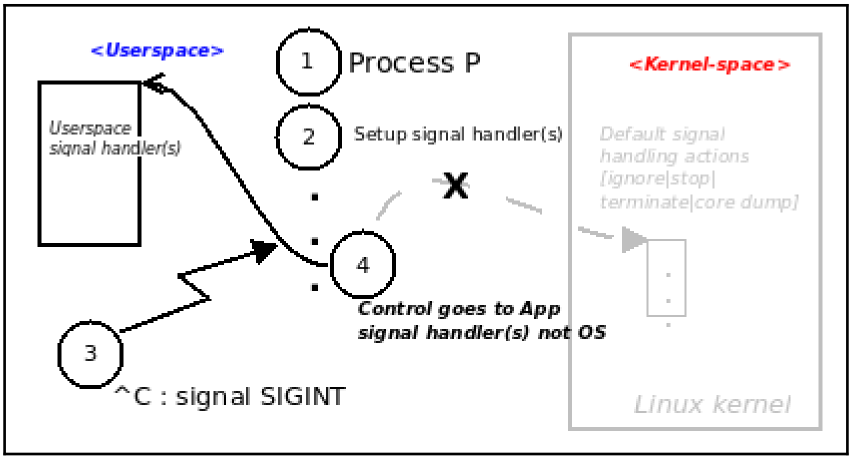


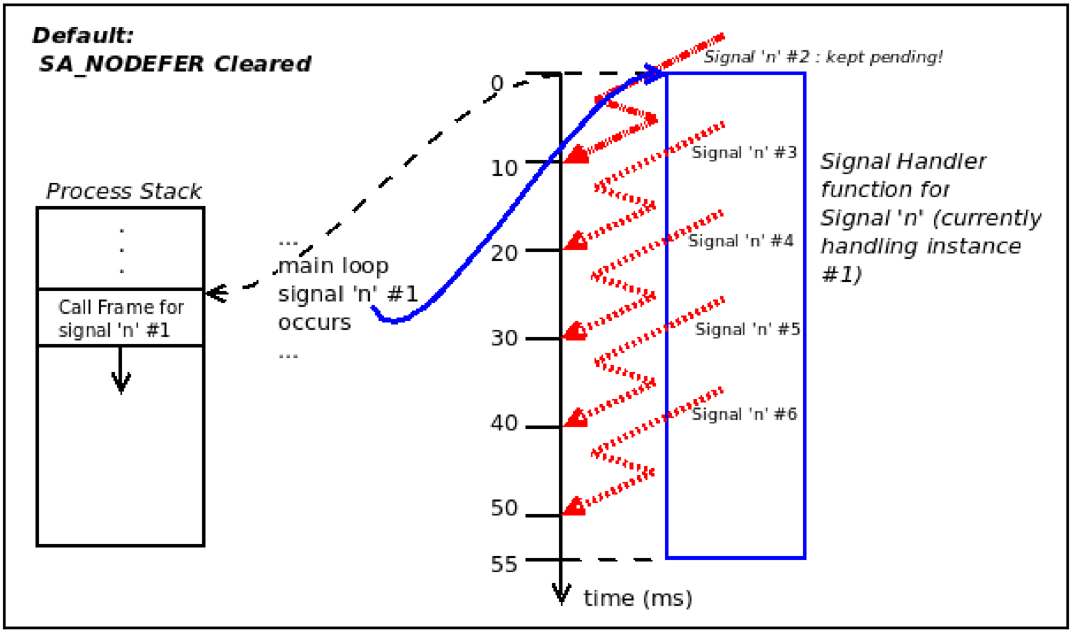



Parent shell
PID x
simpsh
fork()
Parent shell
PID x
simpsh
wait
prompt
simpsh
exec
ps
ps executes
... and dies
Predecessor: simpsh
Child: PID y
Successor: ps
PID y

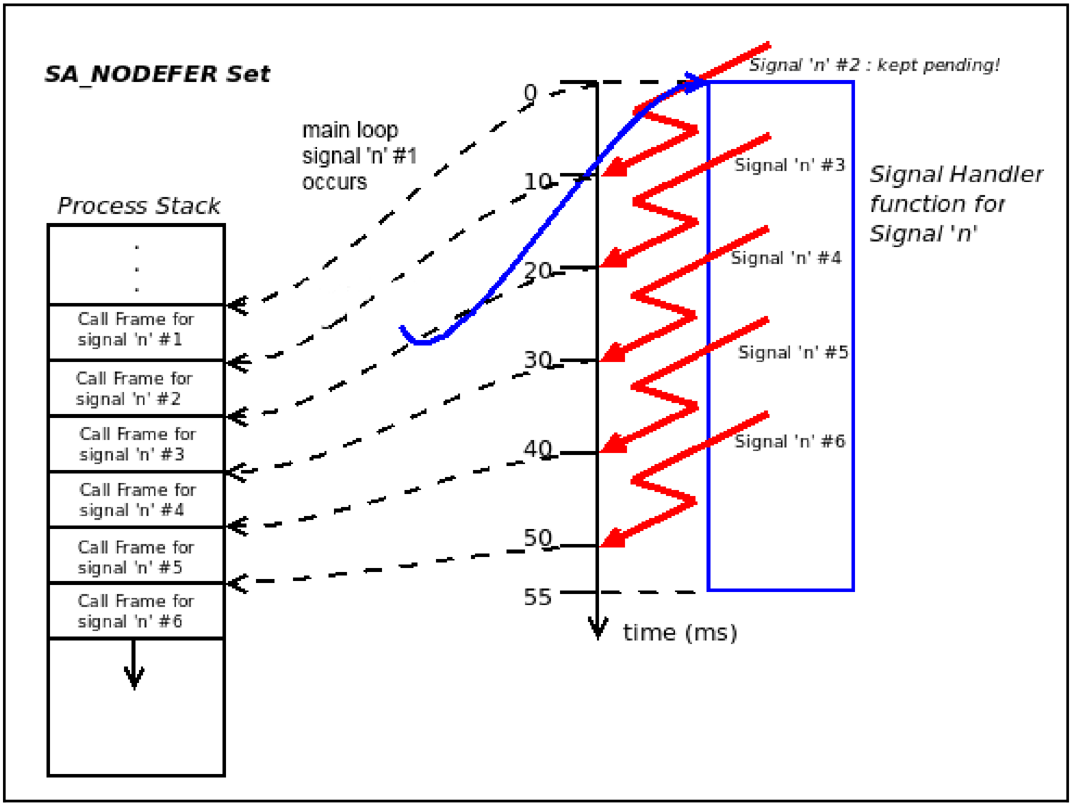
Chapter 11: Signaling - Part I





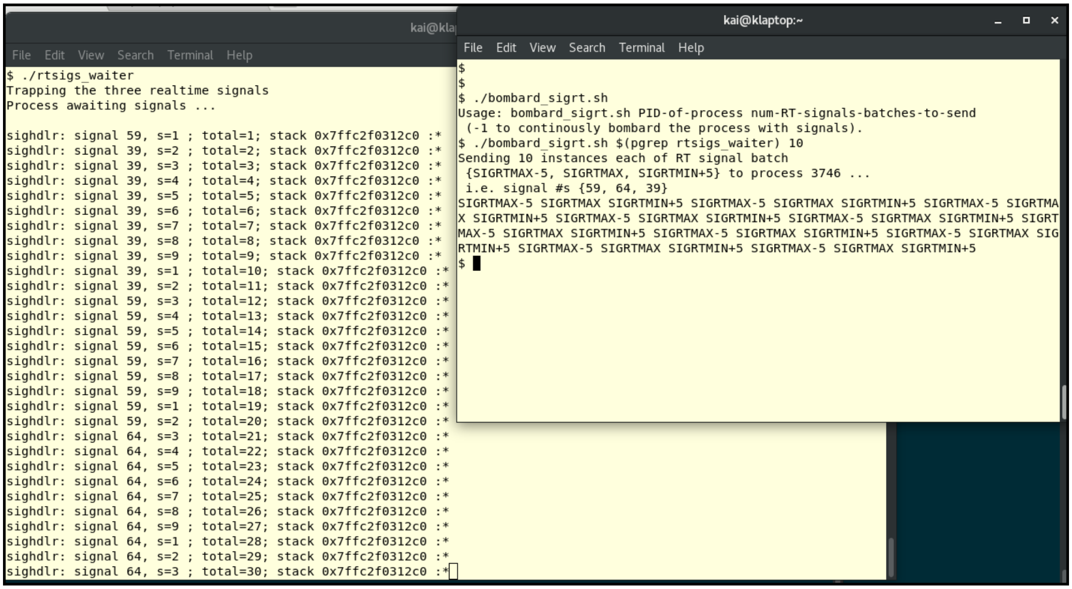


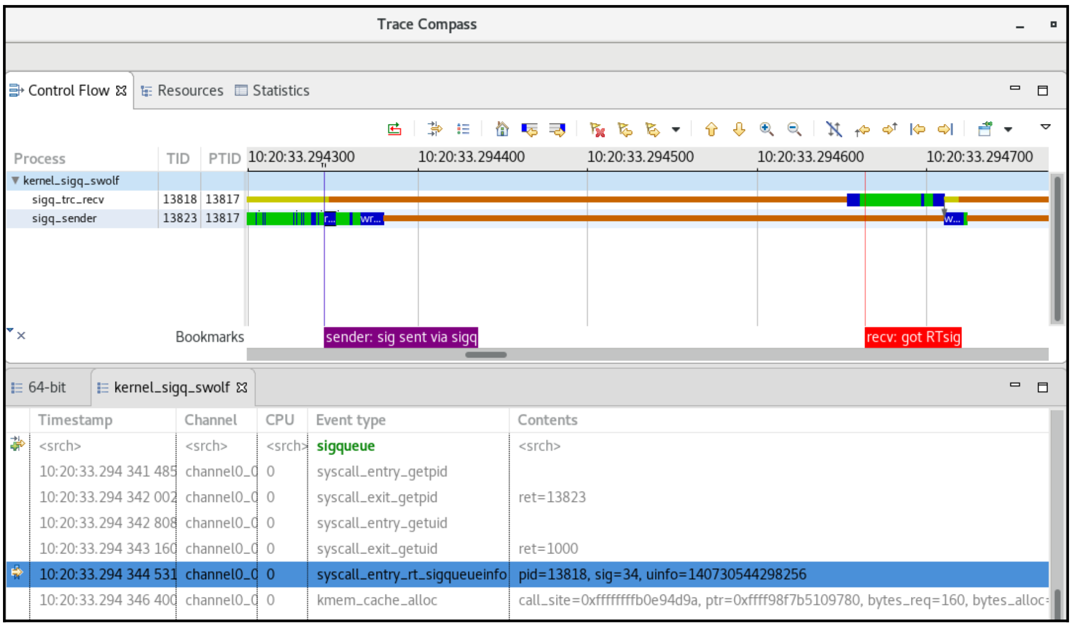
Default:
SA_NODEFER Cleared
Process Stack
Call Frame for
signal 'n' #1
...
main loop
signal 'n' #1
occurs
...
Signal 'n' #2 : kept pending!
Signal 'n' #3
Signal 'n' #4
Signal 'n' #5
Signal 'n' #6
Signal Handler
function for
Signal 'n' (currently
handling instance
#1)
0
10
20
30
40
50
55
time (ms)



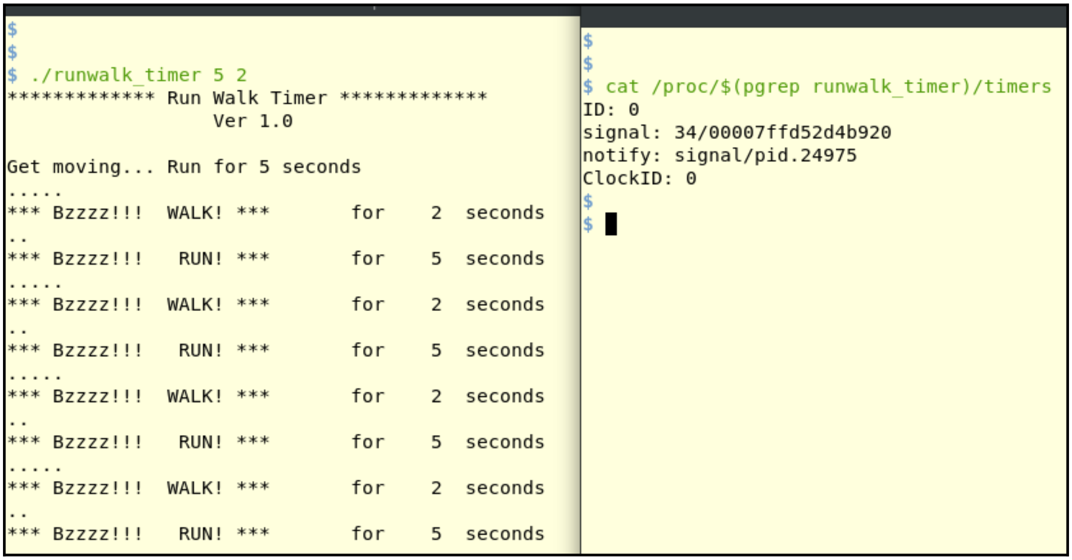
SA_NODEFER Set
Signal 'n' #2 : kept pending!
main loop
signal 'n' #1
occurs
Process Stack
Call Frame for signal 'n' #1
Call Frame for signal 'n' #2
Call Frame for signal 'n' #3
Call Frame for signal 'n' #4
Call Frame for signal 'n' #5
Call Frame for signal 'n' #6
0
10
20
30
40
50
55
time (ms)
Signal 'n' #3
Signal 'n' #4
Signal 'n' #5
Signal 'n' #6
Signal Handler
function for
Signal 'n'

Chapter 12: Signaling - Part II

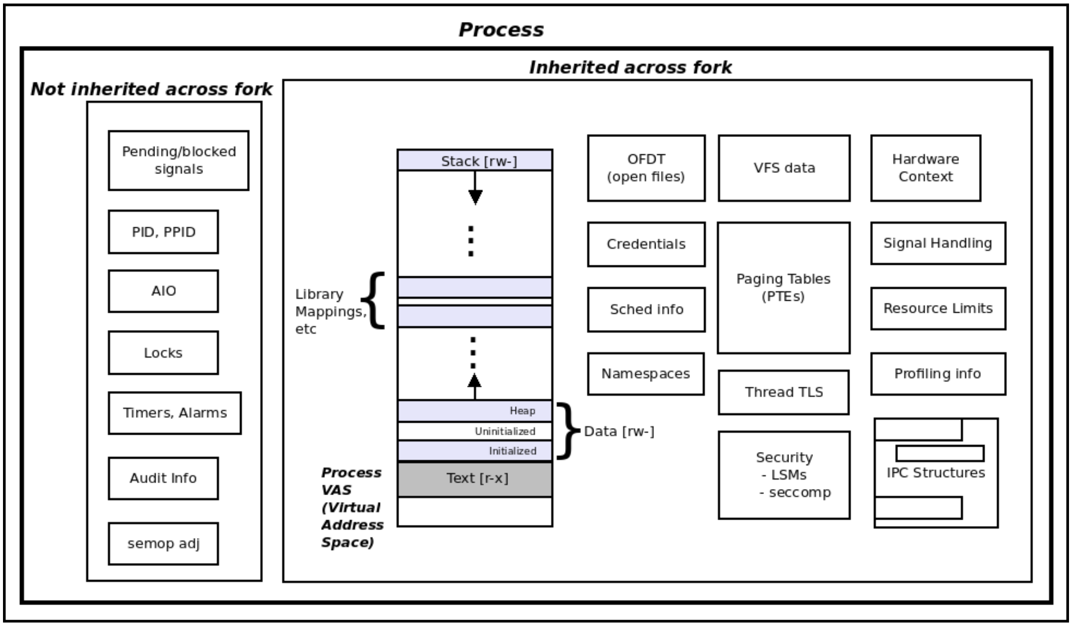


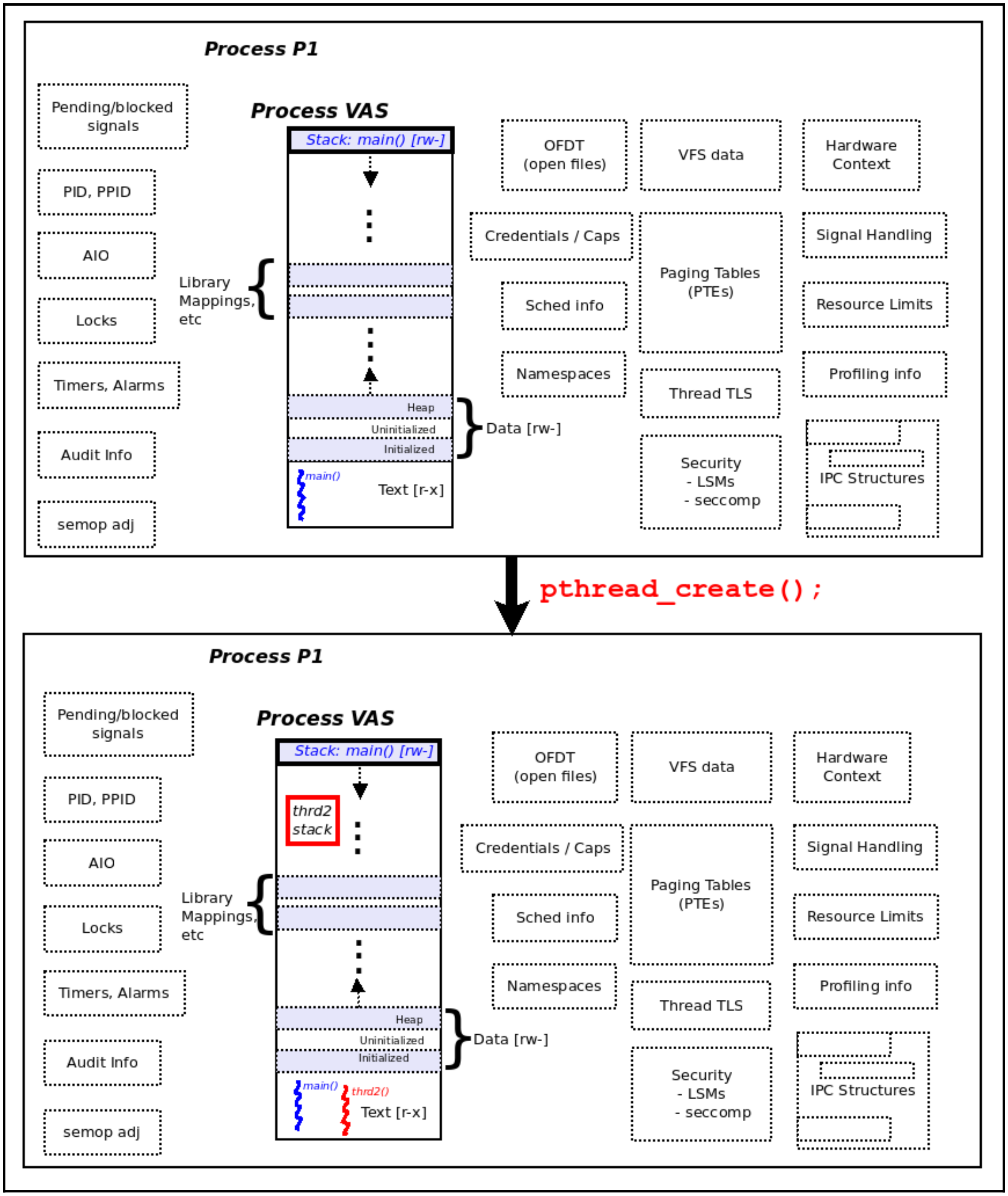


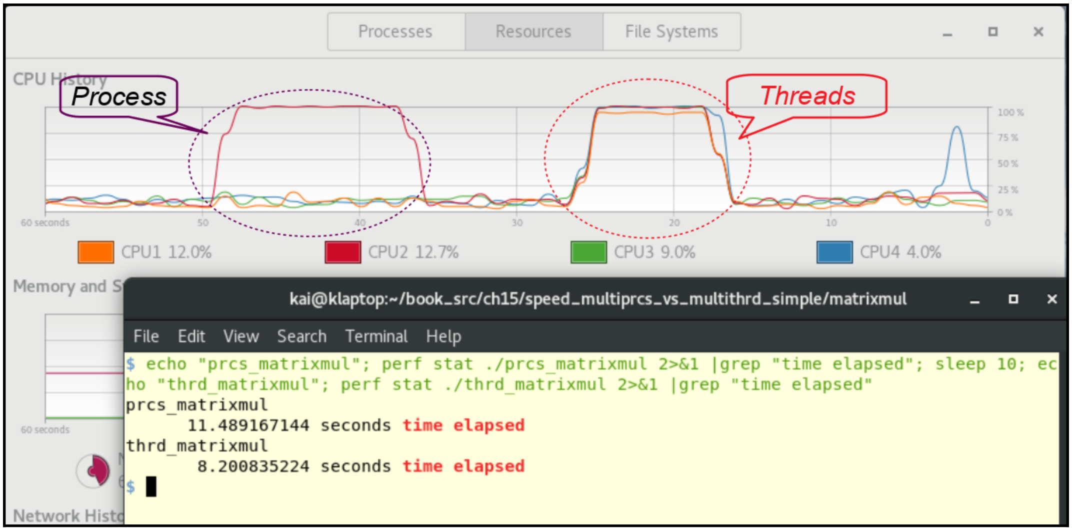
Chapter 13: Timers

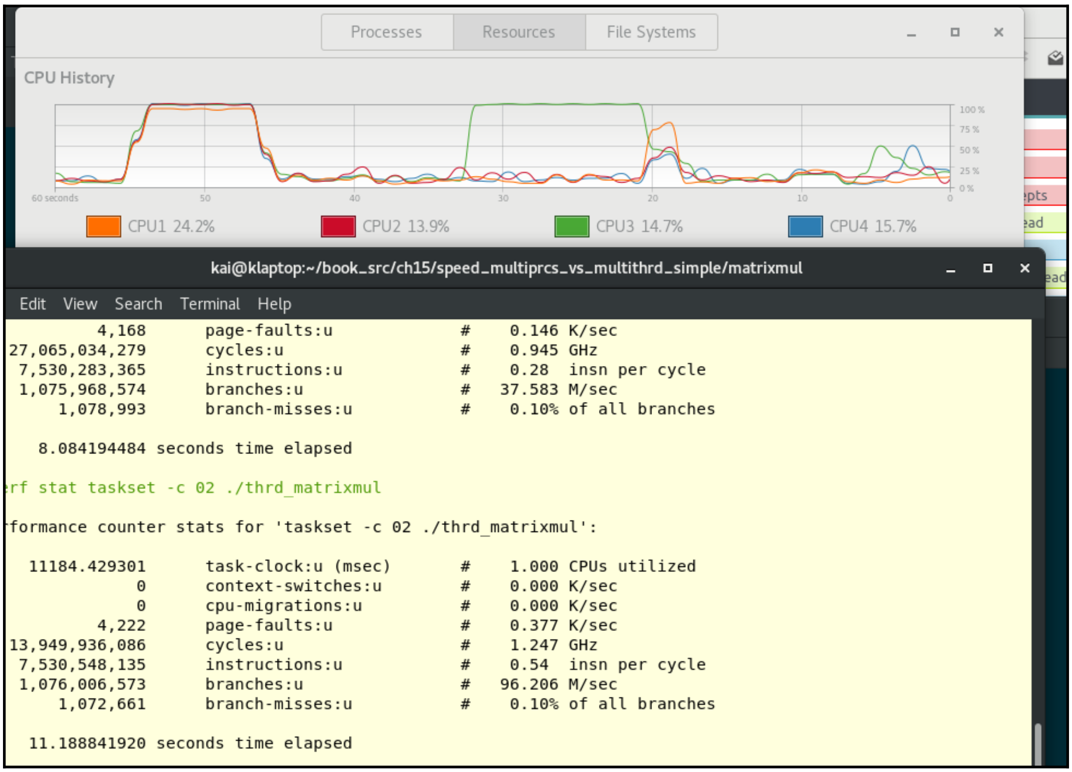


Chapter 14: Multithreading with Pthreads Part I - Essentials

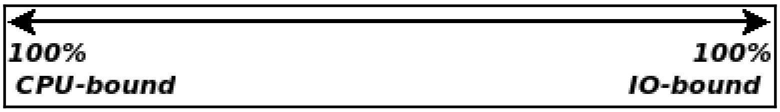



Process P1
Process VAS
Pending/blocked signals
PID, PPID
AIO
Locks
Timers, Alarms
Audit Info
semop adj
Stack: main() [rw-]
Library Mappings, etc
Heap
Uninitialized
Initialized
Data [rw-]
main()
Text [r-x]
OFDT (open files)
VFS data
Hardware Context
Credentials / Caps
Paging Tables (PTEs)
Signal Handling
Sched info
Resource Limits
Namespaces
Thread TLS
Profiling info
Security - LSMs - seccomp
IPC Structures
pthread_create();
Process P1
Process VAS
Pending/blocked signals
PID, PPID
AIO
Locks
Timers, Alarms
Audit Info
semop adj
Stack: main() [rw-]
thrd2 stack
Library Mappings, etc
Heap
Uninitialized
Initialized
Data [rw-]
main()
thrd2()
Text [r-x]
OFDT (open files)
VFS data
Hardware Context
Credentials / Caps
Paging Tables (PTEs)
Signal Handling
Sched info
Resource Limits
Namespaces
Thread TLS
Profiling info
Security - LSMs - seccomp
IPC Structures


Processes
Resources
File Systems
CPU History
Process
Threads
CPU1 12.0%
CPU2 12.7%
CPU3 9.0%
CPU4 4.0%
kai@klaptop:~/book_src/ch15/speed_multiprcs_vs_multithrd_simple/matrixmul
File Edit View Search Terminal Help
$ echo "prcs_matrixmul"; perf stat ./prcs_matrixmul 2>&1 |grep "time elapsed"; sleep 10; ec
ho "thrd_matrixmul"; perf stat ./thrd_matrixmul 2>&1 |grep "time elapsed"
prcs_matrixmul
11.489167144 seconds time elapsed
thrd_matrixmul
8.200835224 seconds time elapsed
$

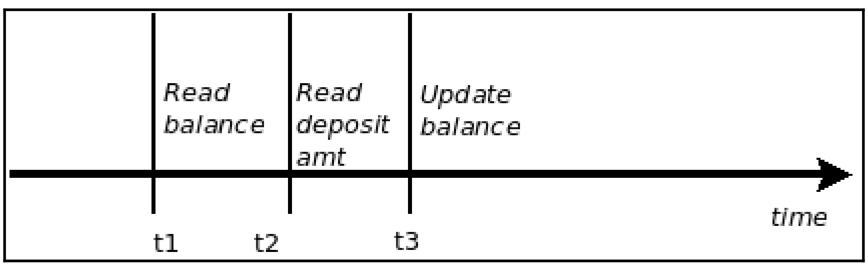

Processes
Resources
File Systems
CPU History
CPU1 24.2%
CPU2 13.9%
CPU3 14.7%
CPU4 15.7%
kai@klaptop:~/book_src/ch15/speed_multiprcs_vs_multithrd_simple/matrixmul
Edit View Search Terminal Help
4,168 page-faults:u # 0.146 K/sec
27,065,034,279 cycles:u # 0.945 GHz
7,530,283,365 instructions:u # 0.28 insn per cycle
1,075,968,574 branches:u # 37.583 M/sec
1,078,993 branch-misses:u # 0.10% of all branches
8.084194484 seconds time elapsed
rf stat taskset -c 02 ./thrd_matrixmul
formance counter stats for 'taskset -c 02 ./thrd_matrixmul':
11184.429301 task-clock:u (msec) # 1.000 CPUs utilized
0 context-switches:u # 0.000 K/sec
0 cpu-migrations:u # 0.000 K/sec
4,222 page-faults:u # 0.377 K/sec
13,949,936,086 cycles:u # 1.247 GHz
7,530,548,135 instructions:u # 0.54 insn per cycle
1,076,006,573 branches:u # 96.206 M/sec
1,072,661 branch-misses:u # 0.10% of all branches
11.188841920 seconds time elapsed


seawolf@seawolf-dev-vm: ~
File Edit View Search Terminal Help
$
$
$ ./pthreads3 &
[1] 3906
$ Worker thread #0 running ...
Worker thread #1 running ...
Worker thread #2 running ...
$ ps -LA|grep pthreads3
3906 3906 pts/0 00:00:00 pthreads3 <defunct>
3906 3907 pts/0 00:00:00 pthreads3
3906 3908 pts/0 00:00:00 pthreads3
3906 3909 pts/0 00:00:00 pthreads3
$ #2: work done, exiting now
#1: work done, exiting now
#0: work done, exiting now

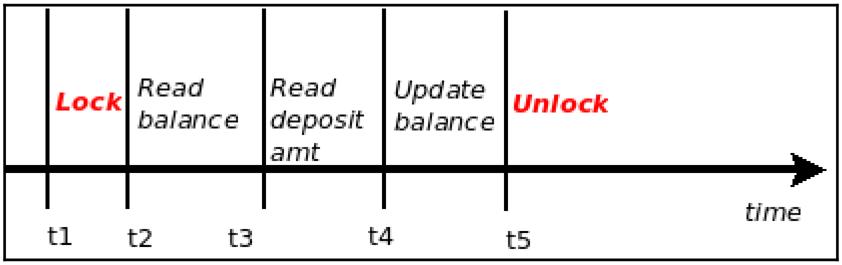

100%
CPU-bound
100%
IO-bound

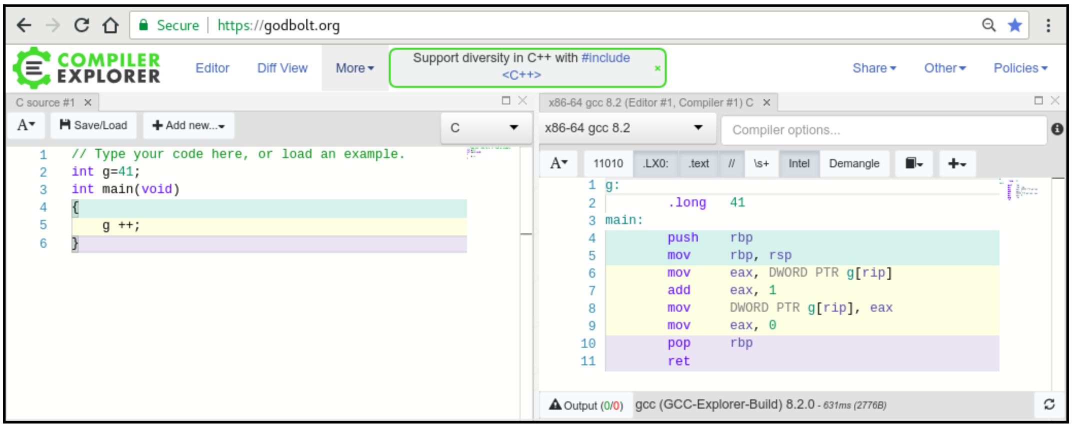
Time	Thread A on CPU 0	Thread B on CPU 1	Balance
t0			12000
t1	read bal		12000
t2		read bal	12000
t3	deposit 3000		12000
t4		deposit 8000	12000
t5	update bal = 12000+3000		15000
t6		update bal = 15000+8000	**23000**

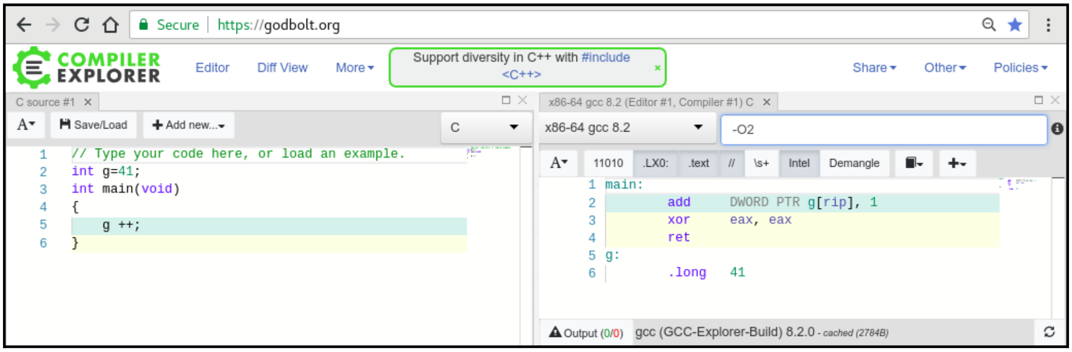
Time	Thread A on CPU 0	Thread B on CPU 1	Balance
t0			12000
t1	read bal		12000
t2		read bal	12000
t3	deposit 3000		12000
t4		deposit 8000	12000
t5	update bal = 12000+3000	update bal = **12000**+8000	15000
t6			**20000**

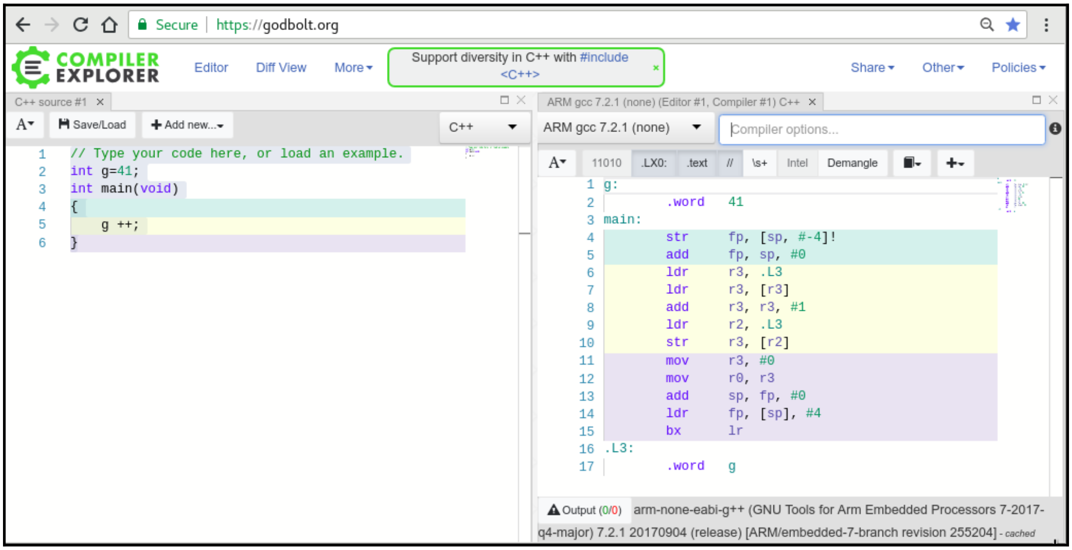


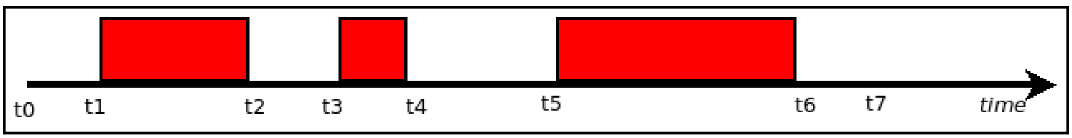
Time	Thread A on CPU 0	Thread B on CPU 1	Balance	
t0			12000	
t1	read bal		12000	Critical Section
t2	deposit 3000		12000	
t3	update bal = 12000+3000		15000	
t4		read bal	15000	Critical Section
t5		deposit 8000	15000	
t6		update bal = 15000+8000	23000	

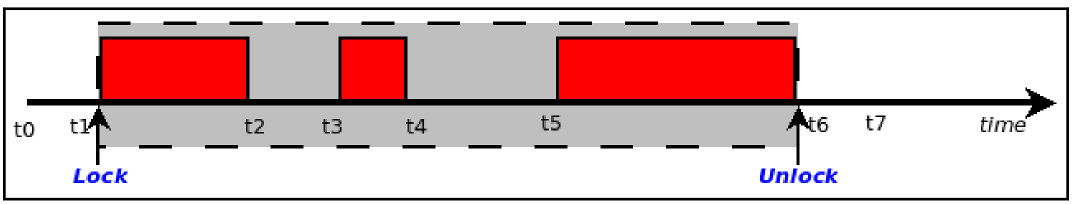


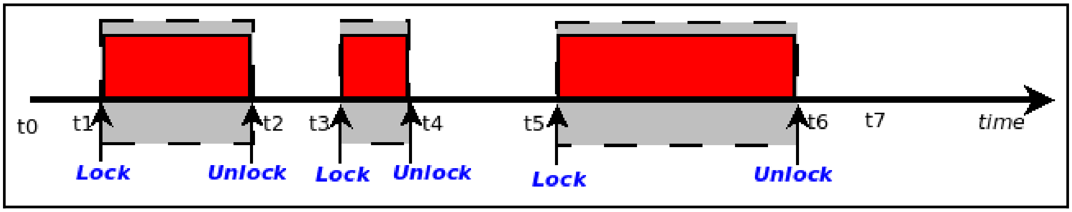



Secure https://godbolt.org
COMPILER EXPLORER
Editor
Diff View
More
Support diversity in C++ with #include <C++>
Share
Other
Policies
C source #1
Save/Load
Add new...
C
// Type your code here, or load an example.
int g=41;
int main(void)
{
g ++;
}
x86-64 gcc 8.2 (Editor #1, Compiler #1) C
x86-64 gcc 8.2
-O2
11010
.LX0:
.text
//
\s+
Intel
Demangle
main:
add DWORD PTR g[rip], 1
xor eax, eax
ret
g:
.long 41
Output (0/0)
gcc (GCC-Explorer-Build) 8.2.0 - cached (2784B)

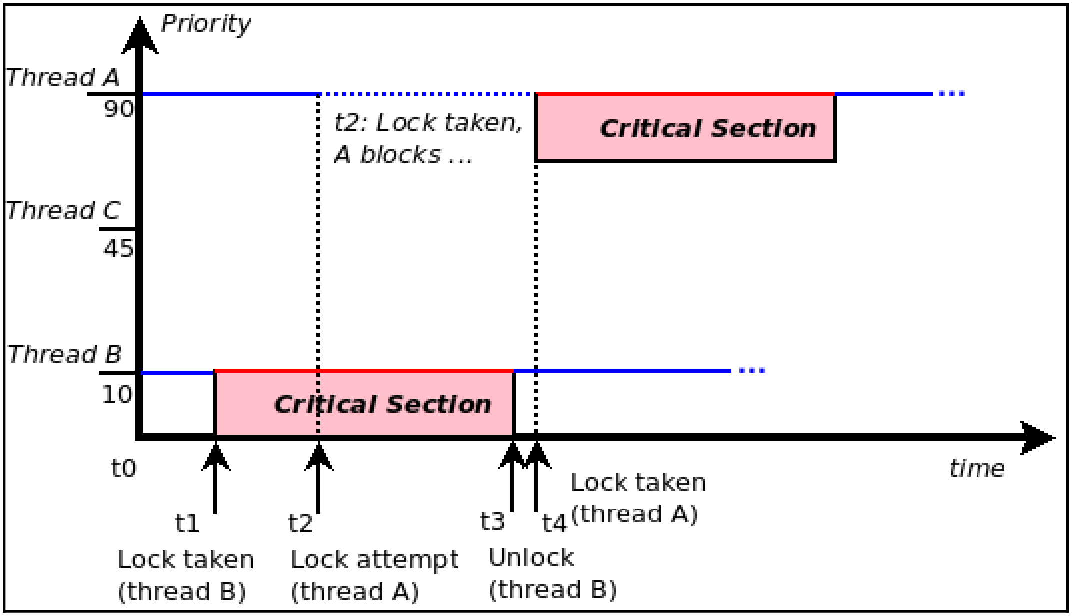

Secure https://godbolt.org
COMPILER EXPLORER
Editor
Diff View
More
Support diversity in C++ with #include <C++>
Share
Other
Policies
C++ source #1
Save/Load
Add new...
C++
// Type your code here, or load an example.
int g=41;
int main(void)
{
g ++;
}
ARM gcc 7.2.1 (none) (Editor #1, Compiler #1) C++
ARM gcc 7.2.1 (none)
Compiler options...
11010
.LX0:
.text
//
\s+
Intel
Demangle
g:
.word 41
main:
str fp, [sp, #-4]!
add fp, sp, #0
ldr r3, .L3
ldr r3, [r3]
add r3, r3, #1
ldr r2, .L3
str r3, [r2]
mov r3, #0
mov r0, r3
add sp, fp, #0
ldr fp, [sp], #4
bx lr
.L3:
.word g
Output (0/0)
arm-none-eabi-g++ (GNU Tools for Arm Embedded Processors 7-2017-q4-major) 7.2.1 20170904 (release) [ARM/embedded-7-branch revision 255204] - cached

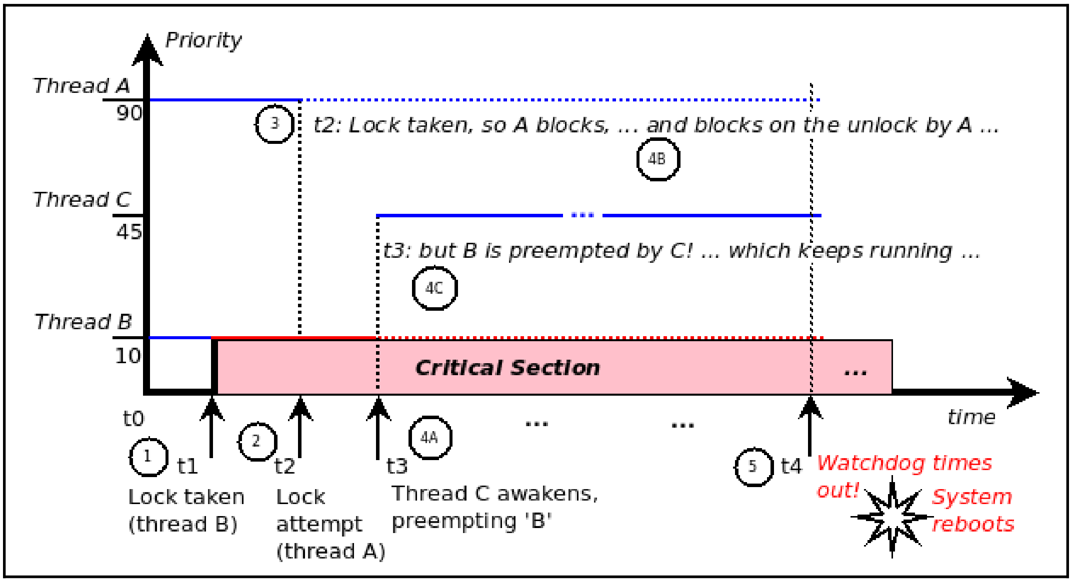

t0
t1
t2
t3
t4
t5
t6
t7
time

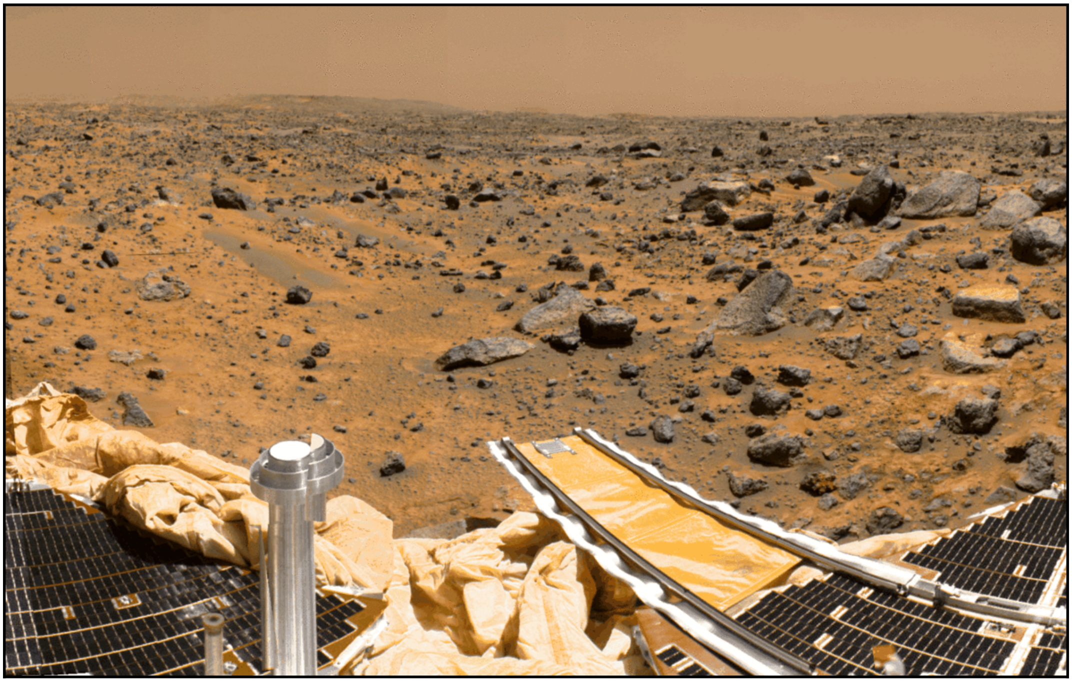

t0
t1
t2
t3
t4
t5
t6
t7
time
Lock
Unlock



Mutex Type	Robustness	Relock	Unlock When Not Owner
NORMAL	non-robust	deadlock	undefined behavior
NORMAL	robust	deadlock	error returned
ERRORCHECK	either	error returned	error returned
RECURSIVE	either	recursive (see below)	error returned
DEFAULT	non-robust	undefined behavior†	undefined behavior†
DEFAULT	robust	undefined behavior†	error returned


Priority
Thread A
90
Thread C
45
Thread B
10
t2: Lock taken,
A blocks ...
Critical Section
Critical Section
t0
t1
t2
t3
t4
time
Lock taken
(thread A)
Lock taken
(thread B)
Lock attempt
(thread A)
Unlock
(thread B)


Priority
Thread A
90
Thread C
45
Thread B
10
t2: Lock taken, so A blocks, ... and blocks on the unlock by A ...
4B
t3: but B is preempted by C! ... which keeps running ...
4C
3
Critical Section
...
t0
1
t1
2
t2
t3
4A
5
t4
time
Lock taken
(thread B)
Lock
attempt
(thread A)
Thread C awakens,
preempting 'B'
Watchdog times
out!
System
reboots

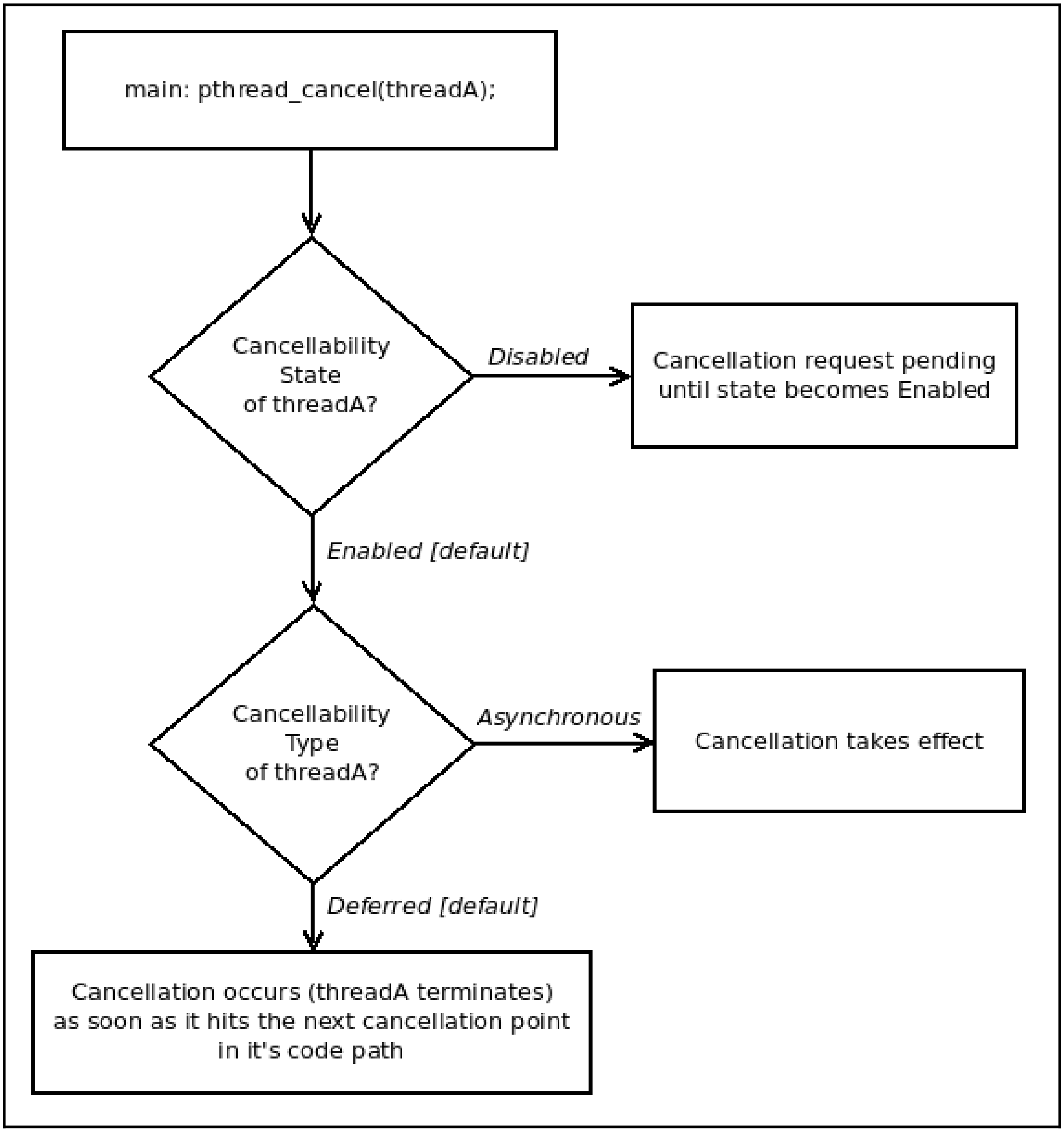


Chapter 16: Multithreading with Pthreads Part III

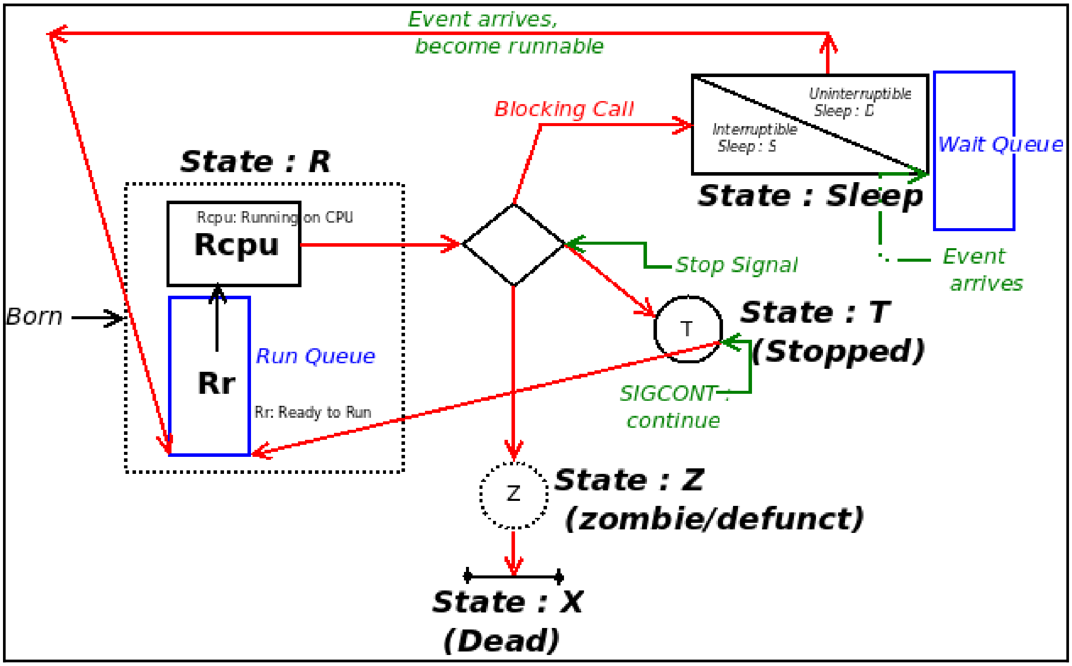
ATTRIBUTES

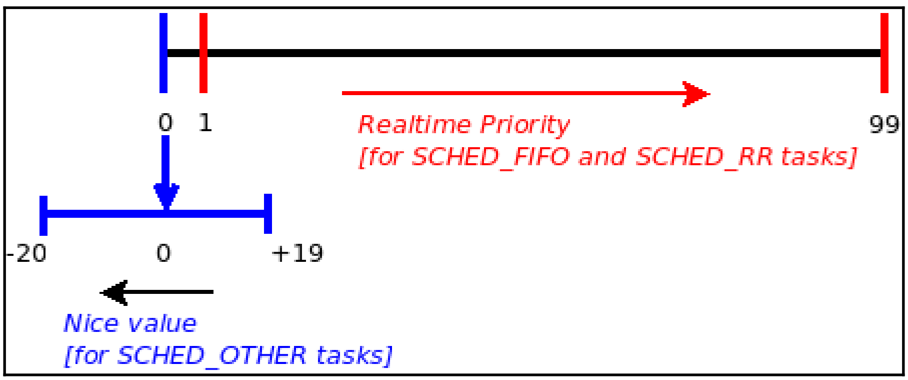
For an explanation of the terms used in this section, see **attributes**(7).

Interface	Attribute	Value
asctime()	Thread safety	MT-Unsafe race:asctime locale
asctime_r()	Thread safety	MT-Safe locale
ctime()	Thread safety	MT-Unsafe race:tmbuf race:asctime env locale
ctime_r(), **gmtime_r**(), **localtime_r**(), **mktime**()	Thread safety	MT-Safe env locale
gmtime(), **localtime**()	Thread safety	MT-Unsafe race:tmbuf env locale

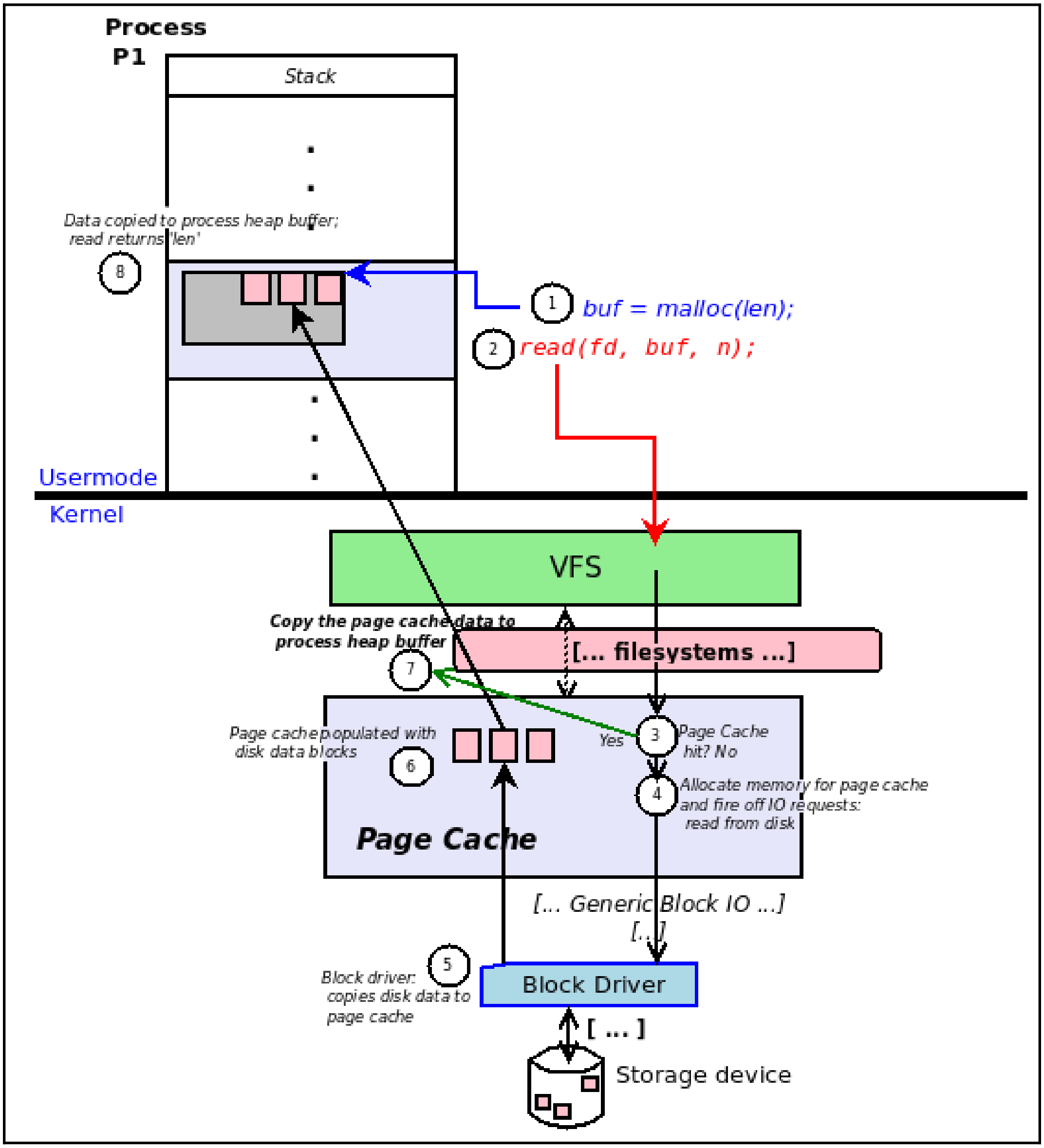

main: pthread_cancel(threadA);
Cancellability
State
of threadA?
Disabled
Cancellation request pending
until state becomes Enabled
Enabled [default]
Cancellability
Type
of threadA?
Asynchronous
Cancellation takes effect
Deferred [default]
Cancellation occurs (threadA terminates)
as soon as it hits the next cancellation point
in it's code path

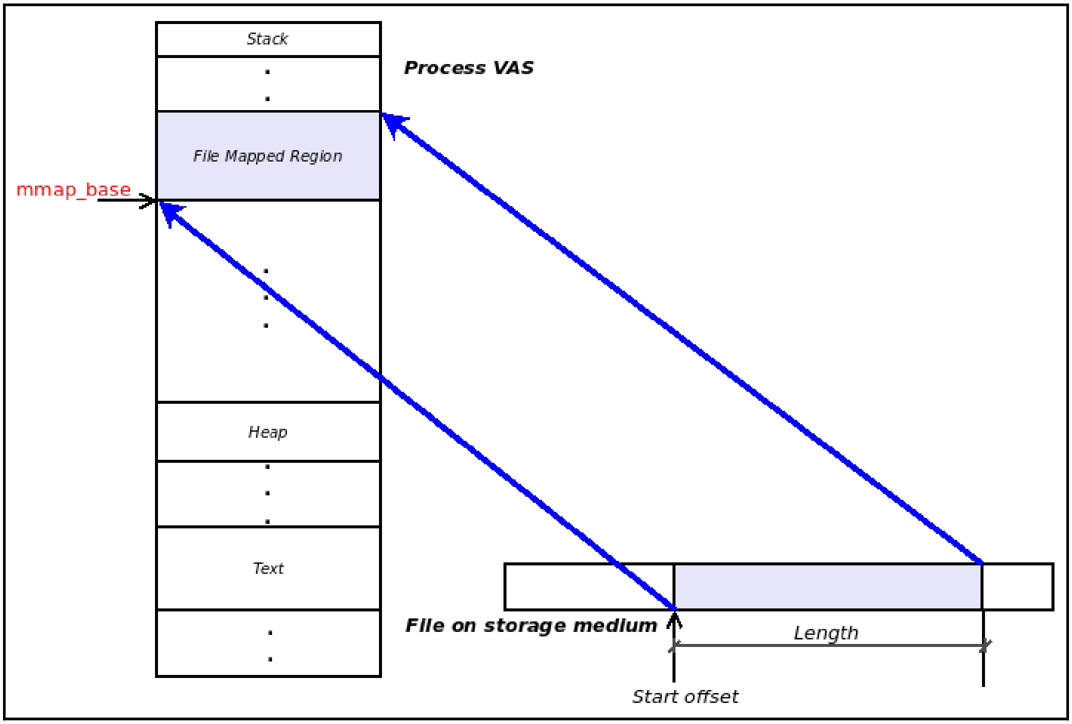
Chapter 17: CPU Scheduling on Linux

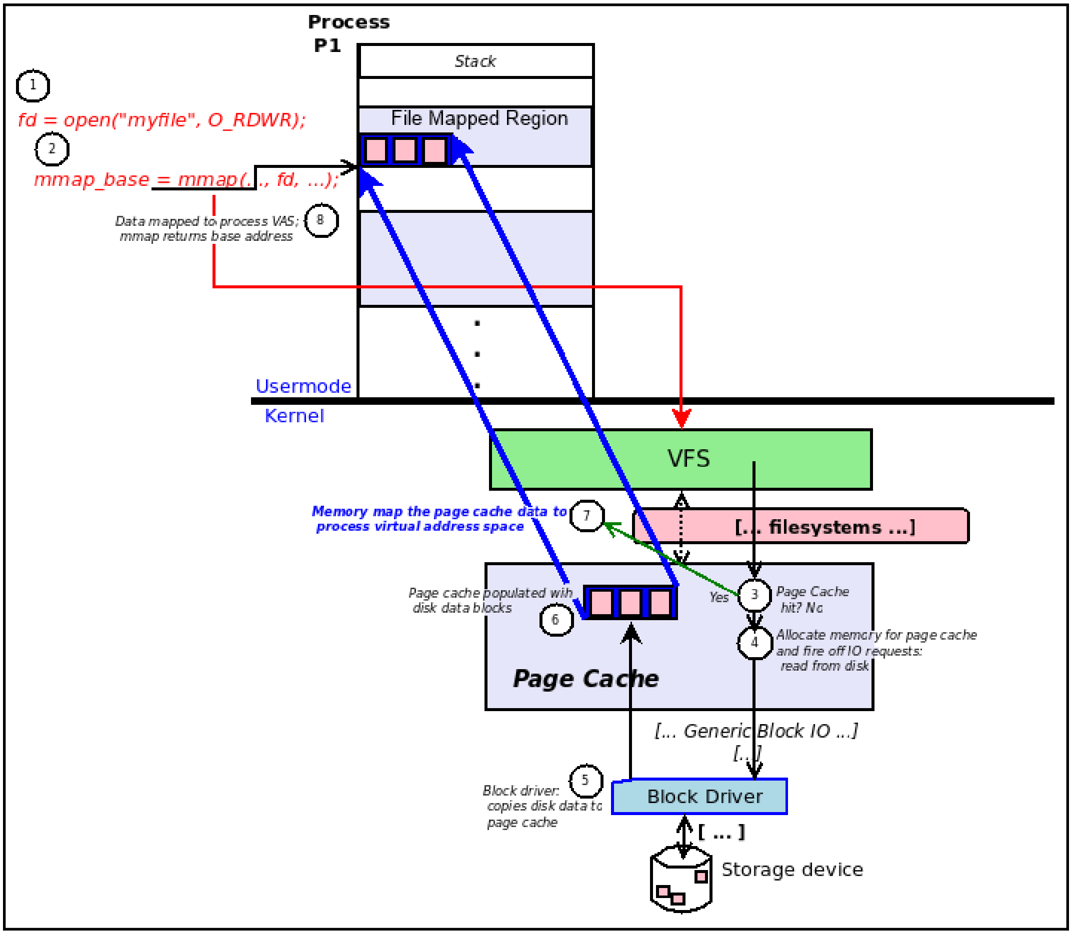


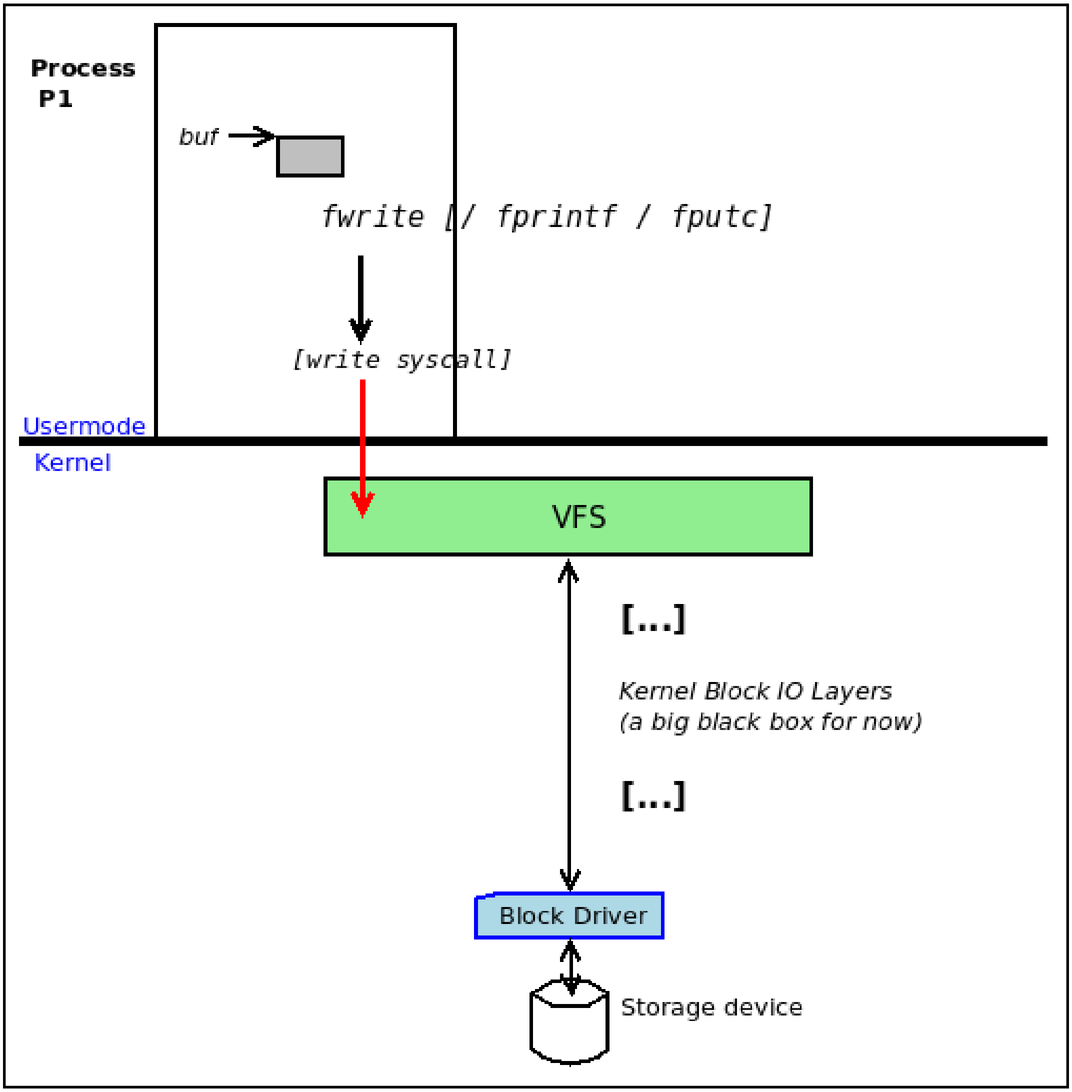


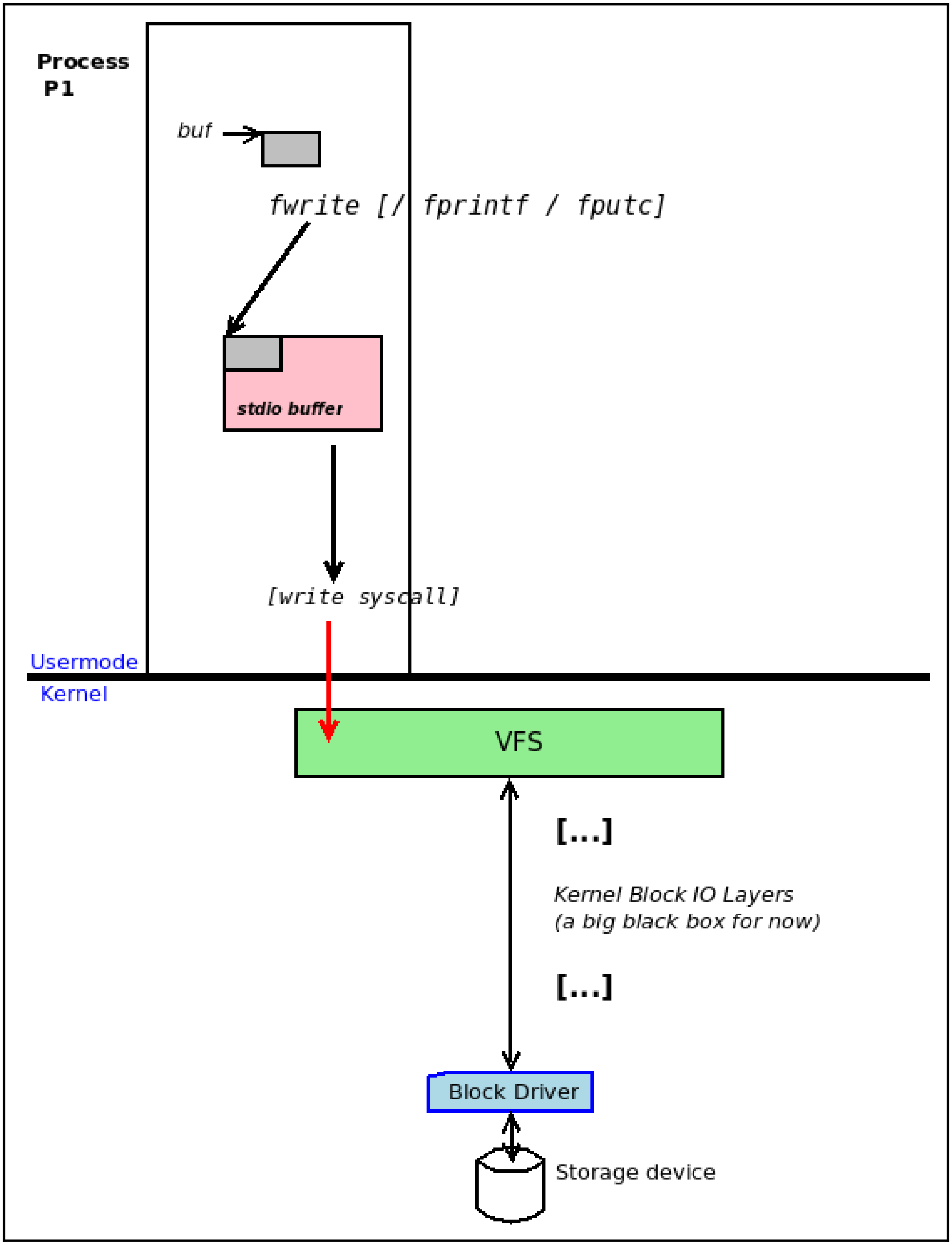
Chapter 18: Advanced File I/O



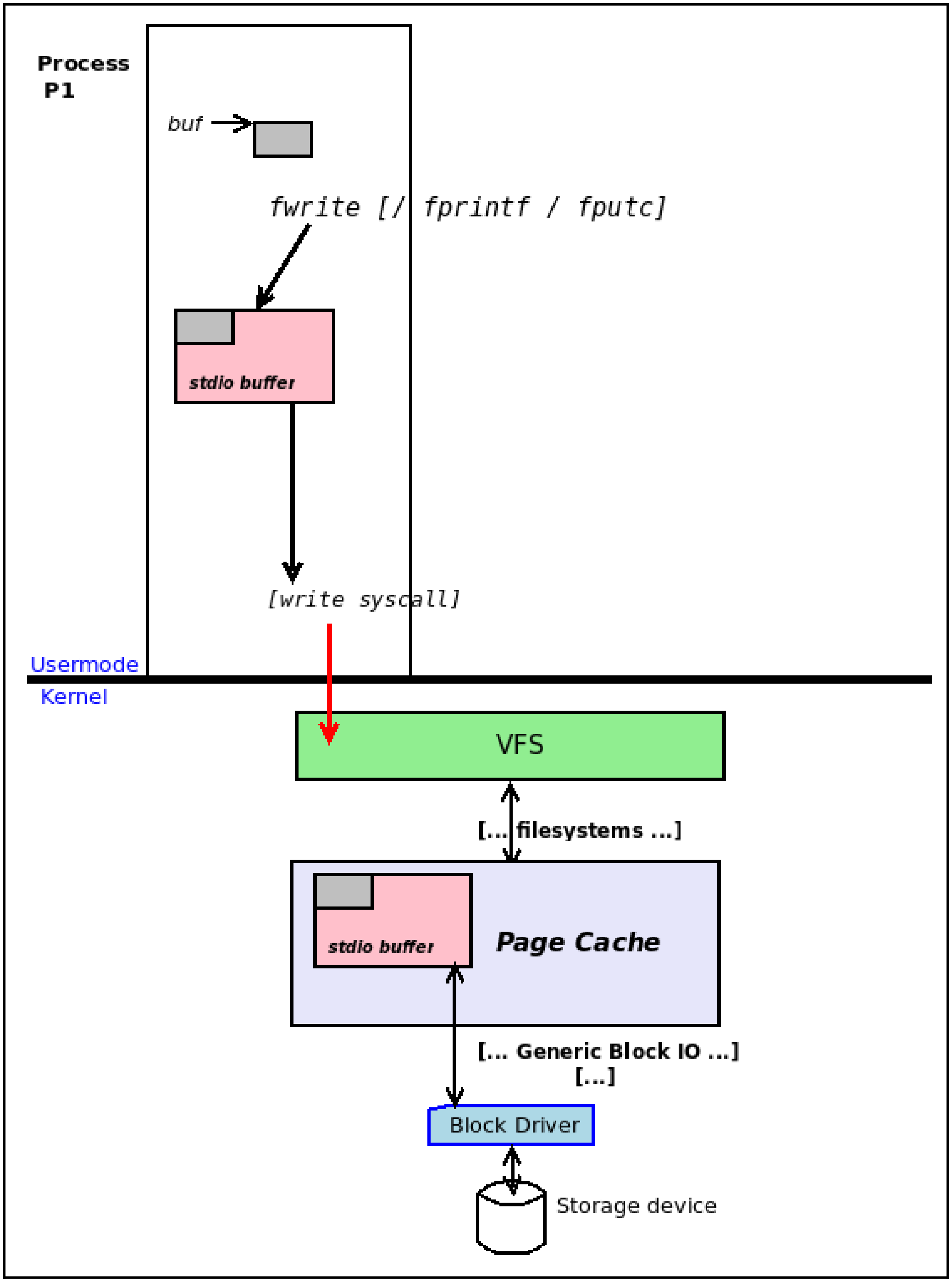

Stack
Process VAS
File Mapped Region
mmap_base
Heap
Text
File on storage medium
Length
Start offset


Process
P1
Stack
File Mapped Region
1
fd = open("myfile", O_RDWR);
2
mmap_base = mmap(..., fd, ...);
Data mapped to process VAS;
mmap returns base address
8
Usermode
Kernel
VFS
Memory map the page cache data to
process virtual address space
7
[... filesystems ...]
Page cache populated with
disk data blocks
6
Yes
3
Page Cache
hit? No
4
Allocate memory for page cache
and fire off IO requests:
read from disk
Page Cache
[... Generic Block IO ...]
[...]
Block driver:
copies disk data to
page cache
5
Block Driver
[...]
Storage device


Process
P1
buf
fwrite [/ fprintf / fputc]
[write syscall]
Usermode
Kernel
VFS
[...]
Kernel Block IO Layers
(a big black box for now)
[...]
Block Driver
Storage device


Process
P1
buf
fwrite [/ fprintf / fputc]
stdio buffer
[write syscall]
Usermode
Kernel
VFS
[...]
Kernel Block IO Layers
(a big black box for now)
[...]
Block Driver
Storage device

```
$
$ make mode_def
gcc -O2 -Wall -UDEBUG -c mode_def.c -o mode_def.o
mode_def.c: In function ‘main’:
mode_def.c:31:7: warning: implicit declaration of function ‘open’; did you mean
‘popen’? [-Wimplicit-function-declaration]
   fd = open(argv[1], O_CREAT|O_RDONLY);
        ^~~~
        popen
mode_def.c:31:21: error: ‘O_CREAT’ undeclared (first use in this function)
   fd = open(argv[1], O_CREAT|O_RDONLY);
                      ^~~~~~~
mode_def.c:31:21: note: each undeclared identifier is reported only once for eac
h function it appears in
mode_def.c:31:29: error: ‘O_RDONLY’ undeclared (first use in this function)
   fd = open(argv[1], O_CREAT|O_RDONLY);
                              ^~~~~~~~
Makefile:113: recipe for target 'mode_def.o' failed
make: *** [mode_def.o] Error 1
$ 
```


Process
P1
buf
fwrite [/ fprintf / fputc]
stdio buffer
[write syscall]
Usermode
Kernel
VFS
[... filesystems ...]
stdio buffer
Page Cache
[... Generic Block IO ...]
[...]
Block Driver
Storage device

Daemon Processes

https://www.freedesktop.org/software/systemd/man/daemon.html

reset.

It is recommended for new-style daemons to implement the following:

1. If `SIGTERM` is received, shut down the daemon and exit cleanly.

2. If `SIGHUP` is received, reload the configuration files, if this applies.

3. Provide a correct exit code from the main daemon process, as this is used by the init system to detect service errors and problems. It is recommended to follow the exit code scheme as defined in the LSB recommendations for SysV init scripts.

4. If possible and applicable, expose the daemon's control interface via the D-Bus IPC system and grab a bus name as last step of initialization.

5. For integration in systemd, provide a `.service` unit file that carries information about starting, stopping and otherwise maintaining the daemon. See systemd.service(5) for details.

6. As much as possible, rely on the init system's functionality to limit the access of the daemon to files, services and other resources, i.e. in the case of systemd, rely on systemd's resource limit control instead of implementing your own, rely on systemd's privilege dropping code instead of implementing it in the daemon, and similar. See systemd.exec(5) for the available controls.

7. If D-Bus is used, make your daemon bus-activatable by supplying a D-Bus service activation configuration file. This has multiple advantages: your daemon may be started lazily on-demand; it may be started in parallel to other daemons requiring it — which maximizes parallelization and boot-up speed; your daemon can be restarted on failure without losing any bus requests, as the bus queues requests for activatable services. See below for details.

8. If your daemon provides services to other local processes or remote clients via a socket, it should be made socket-activatable following the scheme pointed out below. Like D-Bus activation, this enables on-demand starting of services as well as it allows improved parallelization of service start-up. Also, for state-less protocols (such as syslog, DNS), a daemon implementing socket-based activation can be restarted without losing a single request. See below for details.