Debugging

Debugging the Linux Kernel

Debugging kernel problems (or strange user-space interactions) can be hard. There are a variety of techniques that can be applied although first and foremost an understanding of the kernel and what its trying to do is the most useful. Some techniques include:

- Review of the Source Code
- Diagnoses of oops traces
- Tracing system behaviour using the Linux Trace Toolkit
- Debugging using gdb (either via kgdb or an ICE)

See also GDB Scripts

GDBScripts

Here are some useful GDB scripts/macros for debugging the linux kernel. Typically you put these into your ~/.gdbinit file

Viewing Tasks

The following adds the function ps to gdb, allowing you to view the current tasks the kernel knows about:

```
# display running tasks
define ps
        set $initthread = init_tasks[0]
        set $athread = init_tasks[0]
        printf "%d %s\n", $athread->pid, (char*)($athread->comm)
        set $athread = $athread->next_task
        while $athread != ($initthread)
                if ($athread->pid) != (0)
                        printf "%d (%x) %s\n", $athread->pid, $athread->thread.sp, (char*)
$athread->comm
                end
                set $athread = $athread->next_task
        end
end
```

Example usage:

```
(gdb) ps
0 swapper
1 (8822bea4) init
2 (88279f70) keventd
3 (88277f94) ksoftirqd_CPU0
4 (88275f84) kswapd
5 (88273f7c) bdflush
6 (88271f6c) kupdated
8 (882e3f80) rpciod
30 (89dfbea4) syslogd
32 (89dc1f08) klogd
77 (89b6bea4) sshd
78 (88367e90) getty
79 (89d77ea4) sshd
81 (89b35f10) bash
84 (89a67ea4) sshd
86 (899d9f10) bash
(gdb)
```

Getting the task_struct for a given task

```
#
# gettask n
#
# Get the task structure for a given pid
#
define gettask
    set $initthread = init_tasks[0]
    set $athread = init_tasks[0]
    set $athread = $athread->next_task
    while $athread != ($initthread)
        if ($athread->pid) == ($arg0)
            printf "%d %s %x\n", $athread->pid, (char*)$athread->comm,$athread
#           p (struct task_struct) * $athread
            set $gettask = $athread
        end
        set $athread = $athread->next_task
    end
end

define printtask
    gettask $arg0
    p (struct task_struct) * $gettask
end
```

Example:

```
(gdb) printtask 81
81 bash 89b34000
$3 = {state = 1, flags = 256, sigpending = 0, addr_limit = {seg = 2147483648},
  exec_domain = 0x88171544, need_resched = 0, ptrace = 0, lock_depth = -1, counter = 21,
nice = 0,
  policy = 0, mm = 0x8818f220, processor = 0, cpus_runnable = 1, cpus_allowed = 4294967295,
run_list = {
    next = 0x0, prev = 0x0}, sleep_time = 190974, next_task = 0x89a66000, prev_task =
0x89d76000,
  active_mm = 0x8818f220, local_pages = {next = 0x89b34054, prev = 0x89b34054},
allocation_order = 0,
  nr_local_pages = 0, binfmt = 0x8817334c, exit_code = 0, exit_signal = 17, pdeath_signal =
0,
  personality = 8388608, did_exec = -1, task_dumpable = 1, pid = 81, pgrp = 81, tty_old_pgrp
= 0,
  session = 81, tgid = 81, leader = 1, p_opptr = 0x89d76000, p_pptr = 0x89d76000, p_cptr =
0x8999c000,
  p_ysptr = 0x0, p_osptr = 0x0, thread_group = {next = 0x89b340a8, prev = 0x89b340a8},
  pidhash_next = 0x0, pidhash_pprev = 0x88197c34, wait_chldexit = {lock = <incomplete type>,
    task_list = {next = 0x89b35f64, prev = 0x89b35f64}}, vfork_done = 0x0, rt_priority = 0,
  it_real_value = 0, it_prof_value = 0, it_virt_value = 0, it_real_incr = 0, it_prof_incr =
0,
  it_virt_incr = 0, real_timer = {list = {next = 0x0, prev = 0x0}, expires = 189887, data =
2310225920,
    function = 0x88016260 <it_real_fn>}, times = {tms_utime = 31, tms_stime = 20, tms_cutime
= 3,
    tms_cstime = 5}, start_time = 189645, per_cpu_utime = {482}, per_cpu_stime = {1214},
min_flt = 134,
  maj_flt = 321, nswap = 0, cmin_flt = 56, cmaj_flt = 114, cnswap = 0, swappable = -1, uid =
0,
  euid = 0, suid = 0, fsuid = 0, gid = 0, egid = 0, sgid = 0, fsgid = 0, ngroups = 1, groups
= {
    0 <repeats 32 times>}, cap_effective = 4294967039, cap_inheritable = 0, cap_permitted =
4294967039,
  keep_capabilities = 0, user = 0x8817231c, rlim = {{rlim_cur = 4294967295, rlim_max =
4294967295}, {
      rlim_cur = 4294967295, rlim_max = 4294967295}, {rlim_cur = 4294967295, rlim_max =
4294967295}, {
      rlim_cur = 8388608, rlim_max = 4294967295}, {rlim_cur = 0, rlim_max = 4294967295}, {
      rlim_cur = 4294967295, rlim_max = 4294967295}, {rlim_cur = 256, rlim_max = 256}, {
      rlim_cur = 1024, rlim_max = 1024}, {rlim_cur = 4294967295, rlim_max = 4294967295}, {
      rlim_cur = 4294967295, rlim_max = 4294967295}, {rlim_cur = 4294967295, rlim_max =
4294967295}},
  used_math = 1, comm = "bash\0-stop-daem", link_count = 0, total_link_count = 0, tty =
0x89b40000,
  locks = 0, semundo = 0x0, semsleeping = 0x0, thread = {sp = 2310233872, pc = 2281761168,
trap_no = 0,
    error_code = 0, address = 0, ubc_pc1 = 0, ubc_pc2 = 0, fpu = {hard = {fp_regs =
{1073741824, 0,
          1077936128, 0, 1105199104, 0 <repeats 11 times>}, xfp_regs = {0 <repeats 16
times>},
        fpscr = 524292, fpul = 2, status = 0}, soft = {fp_regs = {1073741824, 0, 1077936128,
0,
          1105199104, 0 <repeats 11 times>}, xfp_regs = {0 <repeats 16 times>}, fpscr =
524292,
        fpul = 2, lookahead = 0 '\0', entry_pc = 0} } }, fs = 0x88265220, files =
0x88268520,
  namespace = 0x88234200, sigmask_lock = <incomplete type>, sig = 0x88391580, blocked = {sig
= {0, 0}},
  pending = {head = 0x0, tail = 0x89b3432c, signal = {sig = {0, 0} } }, sas_ss_sp = 0,
sas_ss_size = 0,
  notifier = 0, notifier_data = 0x0, notifier_mask = 0x0, parent_exec_id = 7, self_exec_id =
8,
  alloc_lock = <incomplete type>, journal_info = 0x0}
(gdb)
```