

Operating Systems Lab 1 - Solutions

OS UG TAs

January 2024

Introduction to Linux Tools

Q1

- a. `processor` denotes the index of logical processor while `cores` denote physical cores.
- f. Use `top` or `free` command.
- g. Use `top` or `free` command.
- h. View `/proc/stat` for context switches. Use `vmstat -f` for forks.

Q2

Use the `top` command to view the `pid`. Value under `RES` is the memory being used by the process.

Q3

- `ps aux` or `ps -a` to view processes
- `ps T` to view only of that terminal
- `ps -O pid,ppid,state` to print in a particular format
- `ps -p <pid> -o <headers>` to print for a particular pid

c.

1. `top` for pid
2. `lsuf -p <pid>` to view list of open files with their fd to file name mappings

```
:/proc/6657/fd$ file 0
0: symbolic link to /dev/pts/1
:/proc/6657/fd$ file 1
1: symbolic link to /tmp/tmp.txt
:/proc/6657/fd$ file 2
2: symbolic link to /dev/pts/1
:/proc/6657/fd$ |
```

d.

```

:/proc/1036/fd$ ls
0 1 2
:/proc/1036/fd$ file 0
0: symbolic link to pipe:[29143]

```

e.

```

:/bin$ ls | grep "^ls$"
ls
:/bin$ ls | grep "^cd$"
:/bin$ ls | grep "^history$"
:/bin$ ls | grep "^ps$"
ps

```

Also check `/usr/bin/` for these executables

Q4

```

:/bin$ ps -p 2513 -o pid,comm,size,vsize
PID COMMAND      SIZE  VSZ
2513 memory1      4100  6288

```

```

:/bin$ ps -p 3083 -o pid,comm,size,vsize
PID COMMAND      SIZE  VSZ
3083 memory2      4092  6280

```

VSZ is the total virtual memory, SIZE is the memory actually present in RAM.

Note that WSL gives almost the same output for both the programs but an actual linux PC will give a much lower RES value for memory1 compared to memory2 because the array is never accessed.

Q5

- Use `iostat` after running the programs
- `sudo sync && sudo echo 3 > /proc/sys/vm/drop_caches` for clearing caches
- `echo 3 > /proc/sys/vm/drop_caches`: Writes the value 3 to the `drop_caches` file in the `vm` directory. This value specifies which caches should be cleared (page cache, dentries, and inodes)
- Documentation: <https://www.kernel.org/doc/Documentation/sysctl/vm.txt>

Q6

```

:/bin$ strace -c ls|

```

% time	seconds	usecs/call	calls	errors	syscall
99.91	0.017095	19	874		write
0.09	0.000016	0	26		close
0.00	0.000000	0	9		read
0.00	0.000000	0	25		fstat
0.00	0.000000	0	41		mmap
0.00	0.000000	0	8		mprotect
0.00	0.000000	0	1		munmap
0.00	0.000000	0	5		brk
0.00	0.000000	0	2		rt_sigaction
0.00	0.000000	0	1		rt_sigprocmask
0.00	0.000000	0	2		ioctl
0.00	0.000000	0	8		pread64
0.00	0.000000	0	2	2	access
0.00	0.000000	0	1		mremap
0.00	0.000000	0	1		execve
0.00	0.000000	0	2	2	statfs
0.00	0.000000	0	2	1	arch_prctl
0.00	0.000000	0	1		futex
0.00	0.000000	0	4		getdents64
0.00	0.000000	0	1		set_tid_address
0.00	0.000000	0	24		openat
0.00	0.000000	0	1		set_robust_list
0.00	0.000000	0	1		prlimit64
100.00	0.017111		1042	5	total

Introduction to Debugging Tools

Part A: Debugging with GDB

Few more commands:

- `info breakpoints` - to view information on breakpoints
- `del/dis/en ` - delete/disable/enable a breakpoint
- `step` - to execute a single line of code
- `start` - starts execution of the program, but breaks at the beginning of `main`
- `run <args>` - to run the executable

Q2.

```
(gdb) b fibonacci.cpp:14
Breakpoint 1 at 0x1229: file fibonacci.cpp, line 14.
(gdb) r
Starting program: /mnt/c/Users/Shantanu Welling/Documents/IIT Bombay/CS236/Prev/Lab2/intro-debu
1
1

Breakpoint 1, main (argc=1, argv=0x7fffffffdf08) at fibonacci.cpp:14
14      int next = second_last + last;
(gdb) display last
1: last = 1
(gdb) display second_last
2: second_last = 1
(gdb) c
Continuing.
2

Breakpoint 1, main (argc=1, argv=0x7fffffffdf08) at fibonacci.cpp:14
14      int next = second_last + last;
1: last = 2
2: second_last = 2
(gdb)
Continuing.
4

Breakpoint 1, main (argc=1, argv=0x7fffffffdf08) at fibonacci.cpp:14
14      int next = second_last + last;
1: last = 4
2: second_last = 4
(gdb) i b
```

To rectify, switch lines 16 and 17.

Part B: Memory Check with Valgrind

Sample usage of valgrind on `memory_bugs.c`

```

~/Downloads/stro-debug-code$ valgrind --tool=memcheck --leak-check=yes --show-reachable=yes --num-callers=20 ./membug
==4763== Memcheck, a memory error detector
==4763== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==4763== Using Valgrind 3.18.1 and LibVEX; rerun with -h for copyright info
==4763== Command: ./membug
==4763==
==4763== Syscall param write(buf) points to uninitialised byte(s)
==4763==   at 0x4983697: write (write.c:26)
==4763==   by 0x109235: main (memory_bugs.c:19)
==4763== Address 0x1fffffe0 is on thread 1's stack
==4763==   in frame #1, created by main (memory_bugs.c:9)
==4763==
==4763== Invalid write of size 1
==4763==   at 0x109254: main (memory_bugs.c:26)
==4763== Address 0x4a9a0a0 is 0 bytes inside a block of size 12 free'd
==4763==   at 0x484827f: free (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==4763==   by 0x10924f: main (memory_bugs.c:23)
==4763== Block was alloc'd at
==4763==   at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==4763==   by 0x10923f: main (memory_bugs.c:22)
==4763==
==4763== Invalid read of size 1
==4763==   at 0x10925b: main (memory_bugs.c:29)
==4763== Address 0x4a9a0a0 is 0 bytes inside a block of size 12 free'd
==4763==   at 0x484827f: free (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==4763==   by 0x10924f: main (memory_bugs.c:23)
==4763== Block was alloc'd at
==4763==   at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==4763==   by 0x10923f: main (memory_bugs.c:22)
==4763==
==4763== Invalid free() / delete / delete[] / realloc()
==4763==   at 0x484827f: free (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==4763==   by 0x109290: main (memory_bugs.c:35)
==4763== Address 0x1fffffe0 is on thread 1's stack
==4763==   in frame #1, created by main (memory_bugs.c:9)
==4763==
==4763== HEAP SUMMARY:
==4763==   in use at exit: 80 bytes in 2 blocks
==4763== total heap usage: 4 allocs, 3 frees, 1,116 bytes allocated
==4763==
==4763== 30 bytes in 1 blocks are definitely lost in loss record 1 of 2
==4763==   at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==4763==   by 0x10920d: main (memory_bugs.c:16)
==4763==
==4763== 50 bytes in 1 blocks are definitely lost in loss record 2 of 2
==4763==   at 0x4848899: malloc (in /usr/libexec/valgrind/vgpreload_memcheck-amd64-linux.so)
==4763==   by 0x109280: main (memory_bugs.c:32)
==4763==
==4763== LEAK SUMMARY:
==4763==   definitely lost: 80 bytes in 2 blocks
==4763==   indirectly lost: 0 bytes in 0 blocks
==4763==   possibly lost: 0 bytes in 0 blocks
==4763==   still reachable: 0 bytes in 0 blocks
==4763==   suppressed: 0 bytes in 0 blocks
==4763==
==4763== Use --track-origins=yes to see where uninitialised values come from
==4763== For lists of detected and suppressed errors, rerun with: -s
==4763== ERROR SUMMARY: 6 errors from 6 contexts (suppressed: 0 from 0)

```

memory pointed to by 'p' is never initialized

memory pointed to by 'p' is freed and then 'p' is dereferenced

trying to print '*p'
'p' points to freed memory

using 'free' on statically declared array

4 allocs in total (3 are visible in the code, 1 is inside printf)
only 3 frees