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. `lsof -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 <b>` - 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`