Homework 2 - Operating Systems (ICS431)

Alfaifi, Ammar

1 Commands

1.1 `ps`

This shows processes status. without any options, it will show you a list of processes running in your current terminal session.

1.2 `ps -e`

show all processes, not just those related to the current terminal session

1.3 `ps -u`

will show only the processes started by the current user.

1.4 `ps -el`

command will show the complete list of processes in a long format. It will include the process ID, parent process ID, user ID, group ID, virtual memory size, resident set size, CPU usage, start time, terminal, and command.

2 Zombie Processes

This code is to simulate a zombie process.

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/wait.h>
#include <unistd.h>

int main() {
  pid_t pid;
  pid = fork();

  if (pid < 0) {
    fprintf(stderr, "Fork failed.\n");
    exit(1);
  } else if (pid == 0) {
    printf("This is the child process.\n");
    exit(0);
  } else {
    printf("This is the parent process. Child's PID %d\n", pid);
    sleep(10);
    waitpid(pid, NULL, 0); // wait for child process to exit
    printf("Parent process exiting.\n");
    exit(0);
  }

  return 0;
}
```

From the follwing screenshots, the process with PID of 29296 is the `bash` itself. The running C program has PID of 33017, its parent is the `bash`. When `fork()` is called, a duplicate process from the parent (PID:33017) is created with PID of 33020. This child's PID is returned in the parent process, as seen in the first line in the second figure. When the child is finished but the parent is not, the child's process (33020) becomes a zombie process (see the status Z).

```
ammar-faifi@ammarf:~/Documents$ ps -l
F S   UID    PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1000  29296   29295  0  80   0 -  2262 do_wai pts/0    00:00:00 bash
0 R  1000  32943   29296  0  80   0 -  2452 -      pts/0    00:00:00 ps
ammar-faifi@ammarf:~/Documents$ ./a.out &
[1] 33017
This is the parent process. Child's PID 33020
This is the child process.
ammar-faifi@ammarf:~/Documents$ ps -l
F S   UID    PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1000  29296   29295  0  80   0 -  2262 do_wai pts/0    00:00:00 bash
0 S  1000  33017   29296  0  80   0 -   547 hrtime pts/0    00:00:00 a.out
1 Z  1000  33020   33017  0  80   0 -     0 -      pts/0    00:00:00 a.out <defunct>
0 R  1000  33092   29296  0  80   0 -  2452 -      pts/0    00:00:00 ps
ammar-faifi@ammarf:~/Documents$ Parent process exiting.
ps -l
F S   UID    PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1000  29296   29295  0  80   0 -  2262 do_wai pts/0    00:00:00 bash
0 R  1000  33166   29296  0  80   0 -  2452 -      pts/0    00:00:00 ps
[1]+  Done                    ./a.out
```

3   Multithreaded Program

The codes in C is

```c
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>

void *print_primes(void *arg) {
  int num = *(int *)arg;
  int i, j, flag;
  for (i = 2; i <= num; i++) {
    flag = 1;
    for (j = 2; j <= i / 2; j++) {
      if (i % j == 0) {
        flag = 0;
        break;
      }
    }
    if (flag == 1) {
      printf("%d ", i);
    }
  }
  printf("\n");
  pthread_exit(NULL);
}

int main(int argc, char *argv[]) {
  int num = atoi(argv[1]);

  pthread_t tid;
  pthread_attr_t attr;
  pthread_attr_init(&attr);

  pthread_create(&tid, &attr, print_primes, &num);
  pthread_join(tid, NULL);

  return 0;
}
```

In the following screenshot, I ran the program with an input value of 100. Next I ran it with very large number so I can watch its thread using `ps -lfL` command. I use `>` to pipe the `stdout` to the virtual device `/dev/null` to discard any output, then I use `&` to run it in background.

From the output of `ps`, there are to processes with same PID which indicated it's a multithreaded process. We see also the value of the thread ID (LWP). Also the CPU utilization (C) is large in the thread with value of 93.

4 Proc files

1. By the commnad `cat cpuinfo`, it shows 4 CPU cores in my machine.

2. From `cat meminfo`, I have 4002372 kB ($\sim$ 4 GB), this is true because I'm running VM in my Mac and I specify 4 GB to the Ubuntu VM. And available of 287492 kB (287.49 MB).

```
ammar-faifi@ammarf:/proc$ cat meminfo
MemTotal:         4002372 kB
MemFree:           287492 kB
MemAvailable:     3057608 kB
Buffers:           163580 kB
Cached:           2526304 kB
SwapCached:            92 kB
Active:            995716 kB
Inactive:         2130972 kB
Active(anon):        2208 kB
Inactive(anon):    493472 kB
Active(file):      993508 kB
Inactive(file):   1637500 kB
Unevictable:        61436 kB
Mlocked:            26192 kB
SwapTotal:        2077692 kB
SwapFree:         2076908 kB
Zswap:                  0 kB
Zswapped:               0 kB
Dirty:                  0 kB
Writeback:              0 kB
AnonPages:         498220 kB
Mapped:            232952 kB
Shmem:              58164 kB
KReclaimable:      322592 kB
Slab:              435388 kB
SReclaimable:      322592 kB
SUnreclaim:        112796 kB
KernelStack:         7232 kB
PageTables:         14208 kB
NFS_Unstable:           0 kB
Bounce:                 0 kB
WritebackTmp:           0 kB
CommitLimit:      4078876 kB
Committed_AS:     3603988 kB
VmallocTotal:   133143592960 kB
VmallocUsed:        27568 kB
VmallocChunk:           0 kB
Percpu:              3472 kB
HardwareCorrupted:      0 kB
AnonHugePages:          0 kB
ShmemHugePages:         0 kB
ShmemPmdMapped:         0 kB
FileHugePages:          0 kB
FilePmdMapped:          0 kB
CmaTotal:           32768 kB
CmaFree:             2144 kB
HugePages_Total:        0
HugePages_Free:         0
HugePages_Rsvd:         0
HugePages_Surp:         0
Hugepagesize:        2048 kB
Hugetlb:                0 kB
```

3. From running `cat stat`, then from the line field `ctxt`, It had 5417758 context switches.

```
ammar-faifi@ammarf:/proc$ cat stat
cpu  67149 686 25138 15467377 3354 0 190 0 0 0
cpu0 15037 144 6206 3868406 740 0 72 0 0 0
cpu1 16376 162 6346 3867500 794 0 19 0 0 0
cpu2 16404 180 6194 3867466 854 0 27 0 0 0
cpu3 19332 199 6390 3864004 965 0 71 0 0 0
intr 3904624 0 76320 993156 0 0 0 0 0 0 0 2544663 0 0 0
0 0 0 2743 62360 0 0 0 0 0 27081 34672 30004 33469 0 0 0
ctxt 5417758
btime 1679129575
processes 42381
procs_running 1
procs_blocked 0
softirq 3709491 2357 798097 5 74961 4115 0 63112 1266475
```

4. From the same output I see that my system had forked 42381 processes.

5 Shell Program in C

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/wait.h>
#include <unistd.h>

#define MAX_COMMAND_LENGTH 100
#define MAX_ARGUMENTS 10

int main() {
  char command[MAX_COMMAND_LENGTH];
  char *arguments[MAX_ARGUMENTS];
  pid_t pid;
  int status;
  printf("Enter commands (type 'exit' to quit): \n");
  while (1) {
    printf("201855360> ");
    fgets(command, MAX_COMMAND_LENGTH, stdin);
    // Remove newline character
    command[strlen(command) - 1] = '\0';
    if (strcmp(command, "exit") == 0) {
      printf("Exiting shell program...\n");
      break;
    }
    // Parse command-line arguments
    char *token;
    int i = 0;
    token = strtok(command, " ");
    while (token != NULL) {
      arguments[i++] = token;
      token = strtok(NULL, " ");
    }
    arguments[i] = NULL; // Set last element to NULL for execvp
    pid = fork();
    if (pid == -1) {
      perror("fork");
      exit(EXIT_FAILURE);
    } else if (pid == 0) {
      // Child process
      if (execvp(arguments[0], arguments) == -1) {
        perror("execvp");
        exit(EXIT_FAILURE);
      }
    } else {
      // Parent process
      if (waitpid(pid, &status, 0) == -1) {
        perror("waitpid");
        exit(EXIT_FAILURE);
      }
      printf("Terminated: ");
      if (WIFEXITED(status)) {
        printf("Normally\n");
        printf("Exit status: %d\n", WEXITSTATUS(status));
      } else if (WIFSIGNALED(status)) {
        printf("Due to signal\n");
        printf("Signal number: %d\n", WTERMSIG(status));
      }
    }
  }
  return 0;
}
```

Images for running different commands