בס"ד

תרגיל כיתה 4 – מערכות הפעלה, תשפ"ד

Class Exercise

Part 1

1. Create and compile the program using the code on the next page.
   The compilation command is:
   gcc -pthread -m32 -g targ1.c -o targ1
2. Answer the questions in the code.
3. Using the debugger, set a breakpoint inside the thread code. Which thread does the breakpoint stop in? How can we change this?
4. How many threads are running? When one of the threads hits the breakpoint, what are the other threads running ?
5. While the debugger is stopped at the breakpoint, click on Ctrl-Z. The CLI prompt should open up.
   Run the command ps -Lf
   What does this command show about the process and its threads?

Part 2 (submit screenshot and codes)

See the directions in part 2

Submit all of your answers as a one PDF file(both parts) to the submission box.

Part 1

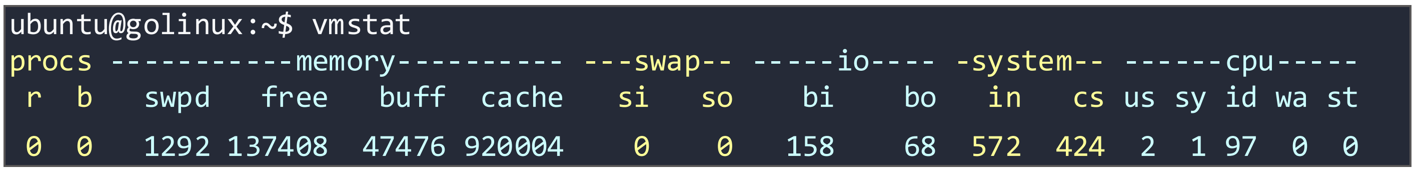
```
#include <stdio.h>
#include <time.h>
#include <sys/types.h>
#include <unistd.h>
#include <pthread.h>
#include <stdlib.h>

void * MyThreadFunction(void * param)
{
    for (int i = 0; i < 10; i++)
    {
        printf("Thread %d %d\n", (int) param, i);
        sleep(1);
     //Q: What does the sleep function do?
    }
    printf("From thread: %d %lu\n",(int) param, pthread_self());
    //Q: What does this function return?
    return (void *) pthread_self();
}

int main(int argc, char** argv)
{
    pthread_t  *ThreadId;
    //Q: What is the size of a pthread_t?
    unsigned int numThreads;
    int retVal;
    pthread_t res;

    srand((unsigned int)time(NULL));
    numThreads = rand() % 3 + 3;
    ThreadId = malloc(numThreads * sizeof(pthread_t));
    for (int i = 0; i < numThreads; i++)
    {
       pthread_create (&ThreadId[i], NULL, &MyThreadFunction, (void *)i);
       //Q: What are the parameters of this function?
      //Q: How can we tell if the thread creation worked?
    }


    for (int i = 0; i < numThreads; i++)
    {

        pthread_join (ThreadId[i],(void*) &res);
        //Q: How can we tell if the thread finished successfully?
        printf("From main: %d %lu\n", i, res);
        //Q: Why is the order of threads different from the printf from within
     the thread (From thread:)
    }

    return 0;
}
```

Part 2

- Process Sate Codes (Meaning of process state codes (C column) on **`ps`** command output):
  Here are the different values that the s, stat and state output specifiers (header "STAT" or "S") will display to describe the state of a process:

  D uninterruptible sleep (usually IO)
  I Idle kernel thread
  R running or runnable (on run queue)
  S interruptible sleep (waiting for an event to complete)
  T stopped by job control signal
  t stopped by debugger during the tracing
  W paging (not valid since the 2.6.xx kernel)
  X dead (should never be seen)
  Z defunct ("zombie") process, terminated but not reaped by its parent
  + is in the foreground process group

- **vmstat** is a powerful and versatile monitoring tool for Linux systems, providing valuable insights into memory, processes, IO, and CPU usage. By understanding the output of vmstat and using it to diagnose potential performance issues, you can optimize your system and ensure that it runs smoothly and efficiently.



```
ubuntu@golinux:~$ vmstat
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 0  0   1292 137408  47476 920004    0    0   158    68  572  424  2  1 97  0  0
```

FIELD DESCRIPTION FOR VM MODE

Procs

- **r:** The number of **runnable** processes (running or waiting for run time).
- **b:** The number of processes **blocked** waiting for I/O to complete. צוואר בקבוק של המעבד b>0

System

- **in**: The number of **interrupts** per second, including the clock.
- **cs**: The number of **context switches** per second.

CPU

These are percentages of total CPU time.

- **us**: Time spent running non-kernel code. (**user time**, including nice time)
- **sy**: Time spent running kernel code. (**system time**)
- **id**: Time spent **idle**.
- **wa**: Time spent **waiting for IO**. צוואר בקבוק של מאגר הנתונים wa>0
- **st**: Time stolen from a virtual machine.
- **gu**: Time spent running KVM guest code (guest time, including guest nice).

הפקודה ps –l נותנת פלט במבנה הבא:

```
myghaz@acad.jct.ac.il@linapp-1:~/OS$ ps -l
F  S  UID           PID       PPID      C PRI  NI ADDR  SZ    WCHAN  TTY      TIME      CMD
0  S 1437018013 3614011 3614001  0  80   0    -      2451 do_wait  pts/35   00:00:00  bash
0  R 1437018013 3622825 3614011  0  80   0    -      1716  -        pts/35   00:00:00  ps
```

כתבו תכנית בשפת C/C++ המריצה **מתוכה** את הפקודה ps –l (השתמשו ב- execl) ואז בעזרת צינור | (pipe) תעבירו את הפלט לתוכנית אחרת וספרו כמה תהליכים מיתוך מה שקיבלתם הם מצב מוכנים לריצה, העבירו בעזרת צינור | (pipe) לתוכנית שלישית והדפיסו כמה אחוזים מתוך כלל התהליכים המוכנים לרוץ המופיעים בפקודה vmstat הם התהליכים שלכם?

דרך פתרון:

./prog1 | ./prog2 | ./prog3

prog1 תפעיל ב- execl את ps –l.

prog2 תשלוף את השדה בעמודה השניה ותספור כמה מהם הם R תדפיס(למסך, למעשה לצינור) את המספר ואח"כ תפעיל ב- execl את vmstat.

prog3 תיקרא מה- pipe את השורה הראשונה (מספר התהליכים שאנו הרצנו) ותשלוף מהשורה הרביעית(תבינו לאיזה פלט מתייחס מספר השורה) את השדה הראשון ותבצע חלוקה של המספרים ותדפיס למסך את הפלט הבא: **Part 2:** < result here >

הערה: לפעמים פלט vmstat מראה 0 תהליכים RUNNABLE , אם כן, תריצו את הכל כמה פעמים.

Running vmstat 3 times every 1 second, example only

```
ubuntu@golinux:~$ vmstat 1 3
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
 r  b   swpd   free   buff  cache   si   so    bi    bo   in   cs us sy id wa st
 0  0   1292 137408  47476 920004    0    0   158    68  572  424  2  1 97  0  0
 2  0   1292 137412  47476 920000    0    0   158    68  572  424  2  1 97  0  0
 0  0   1292 137408  47476 920004    0    0   158    68  572  424  2  1 97  0  0
```

Examples to use vmstat command

- **-a, --active** : active/inactive memory
- **-f, --forks** : number of forks since boot
- **-m, --slabs** : slabinfo
- **-s, --stats** : event counter statistics
- **-d, --disk** : disk statistics

```
$ vmstat -a
```

```
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu-----
r  b   swpd   free  inact active   si   so    bi    bo   in   cs  us sy id wa st
1  0    0     7648  4449  10561    0    3     0     12    1    1   16  1 83  0  0
```

```
$ vmstat -f
```

```
12623277 forks
```