

ITSC205: Operating Systems Internals

ITSC205: Operating Systems Internals

Lab QUIZ #1

**NAME:\_\_Coleton Sanheim\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Mark:\_\_\_\_\_\_/42**

***Read each question carefully, and attach requested screen captures to demo results in this document. Attach screen capture just below the respective question. You are allowed to use windows and Linux virtual machines and labs***

*“Academic dishonesty in any fashion is a serious offence.  Anyone caught cheating will be dealt with according to SAIT’s academic policy and procedure, Student Code of Conduct AC 3.4 and AC 3.4.1, and as has been detailed in the ETHI 110 Academic Honesty Awareness Tutorial.”*

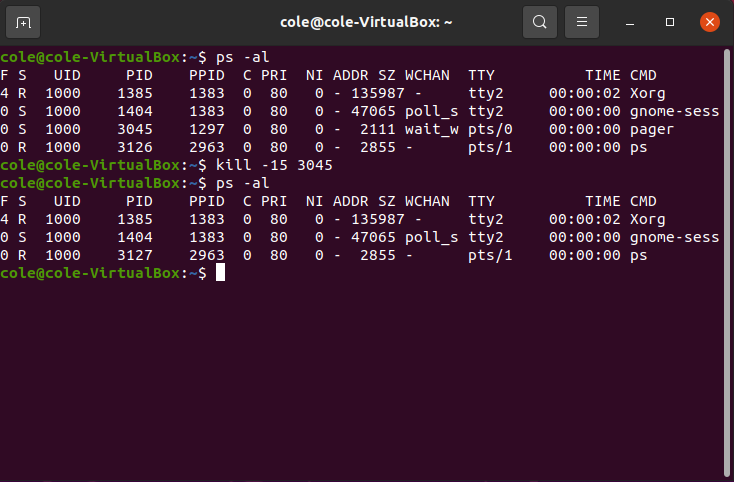
**Linux Operating System**  **\_\_\_\_/25**

1. (3 marks) Create a zombie process in one terminal and display ***the* zombie process attributes.** From a second terminal send the respective signal to terminate the zombie process**.** Attach screen captures that demo the following:
   1. Zombie process creation and display respective attributes (pid,state,name)

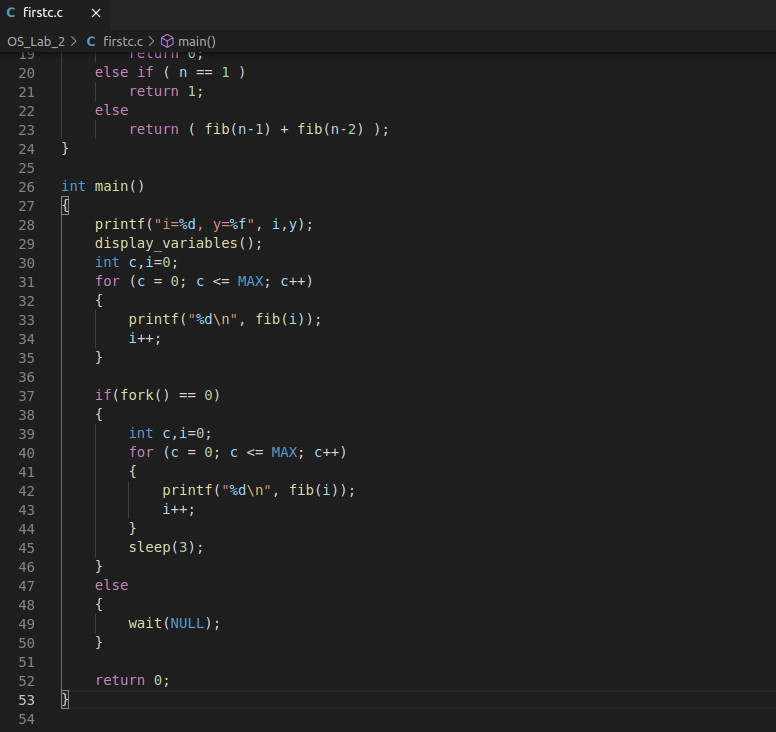
Graphical user interface

Description automatically generated with medium confidence

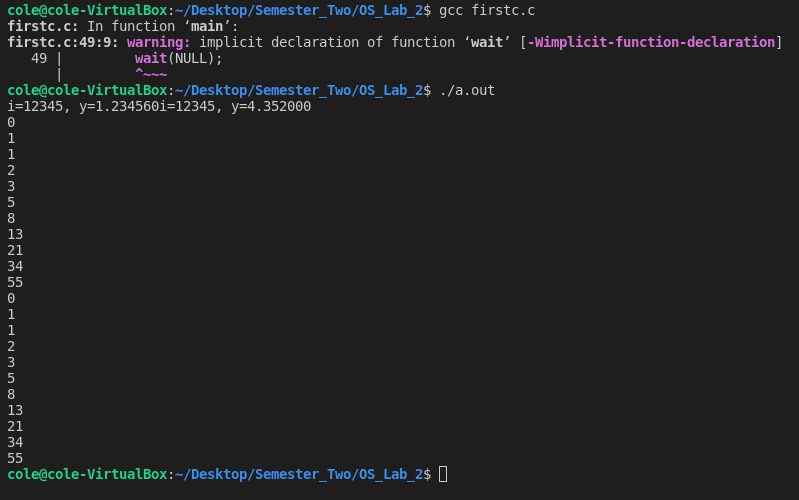
* 1. Second terminal sending Terminate signal to the zombie process



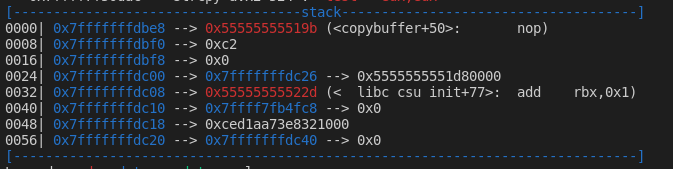
1. (4 marks) Modify the code first.c (program that calculates Fibonacci series, created in Lab 2 Section 4.0) in a way that:
   1. Clones a child process and the child process will calculate Fibonacci and sleep for 3 seconds
   2. Parent should wait(NULL)
   3. Attach screen captures that demo the following:
      1. Modified code



* + 1. Results after compiling and running the code



1. (4 marks) Compile stackover.c program with the respective options to debug the code. Use gdb (peda or gef is OK) to create breakpoints at ***main*** and ***copybuffer*** functions and run the program with 10 (bs) bbbbbbbbbb. Press n (next) many times to run many cycles of the program. Attach the screen captures that demo the following:
   1. The **stack** of this program



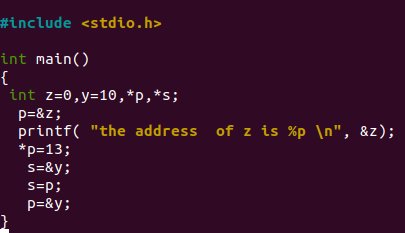
* 1. The **stack base pointer** , **stack pointer** and **RIP** addresses of this program



* 1. Run the program again and generate stack buffer overflow. Display the results after overflowing the stack(registers values, stack and Abort or Seg Fault signal)



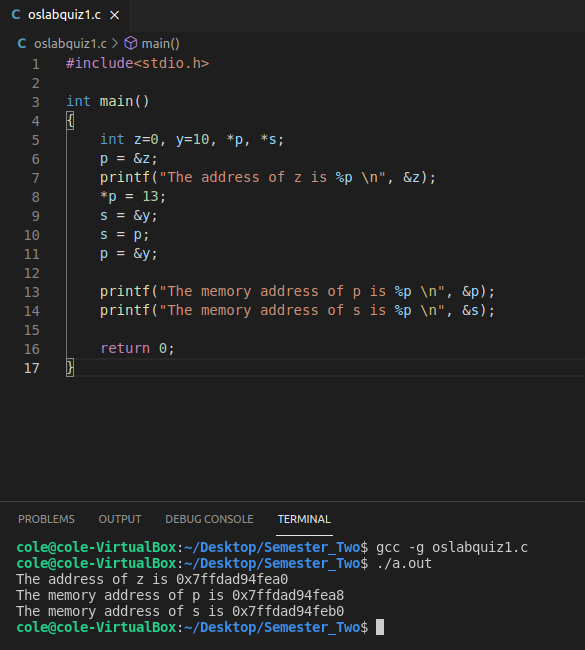
1. (6 marks) Type the following code:



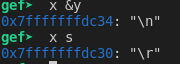
1. Add to the code the respective printf function to print the memory address of

pointers p and s. Compile and run the code. Attach a screen capture with

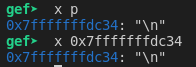
modified code and results after compilation.



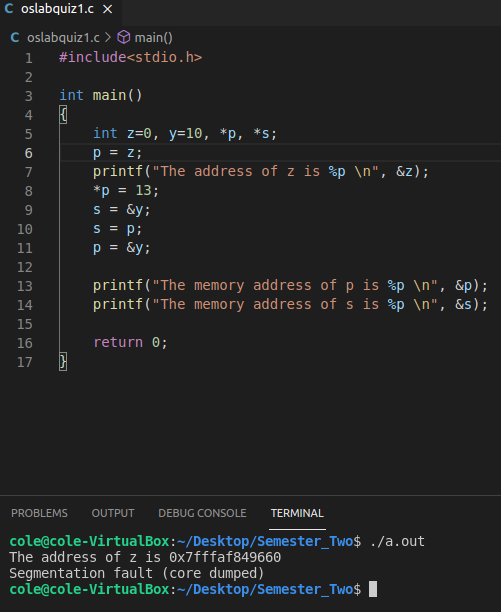
1. Use gdb to debug the program. Attach a screen capture that demos how to use gdb command to display the **memory content** of y and s



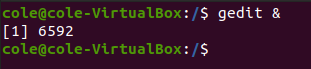
1. Use **x** command with the address of ***p*** to display the content of ***p***

******

1. Attach a screen capture that demos **segmentation fault** caused by pointer error (**s** or **p**).



1. (5 marks) Start ***nano*** or **gedit** process. Run the process in the background
   1. Use a the respective command to find the PID of this process
   2. Use /proc to find the **base** virtual addresses of the **text section** of this process
   3. Use **pagatables.c** program to find the physical address of one of the base address identified for text section for this process
   4. Attach the screen captures that demos
      1. Nano or gedit process running in the background



* + 1. Identified virtual based addresses for **text section** **only**

****

* + 1. The respective physical address of one of identified virtual based addresses



1. (3 marks) Provide Linux commands that will display the following:
   1. Page size

**getconf PAGESIZE**

* 1. All processes states

**ps -al**

**ps aux**

* 1. Signals names

**kill -l**

* 1. Memory information

**top**

* 1. System calls of a running process

**strace**

* 1. Content of a variable

**printenv <variable>**

**Windows Operating System**   **\_\_/17**

1. (2 marks) The latest Windows versions implemented secure boot. List two ways to verify if secure boot is enable or not.
   1. **bcdedit**
   2. **msconfig**
2. ( 2 marks) What Windows command will display:
   1. Computer name

**hostname**

* 1. Content of **PATH** variable

**echo %path%**

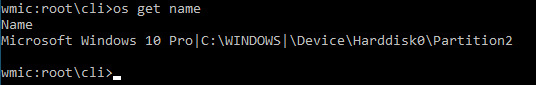
1. ( 3 marks) Use the respective Windows tool that displays process creation and the first DLL loaded after the process is created and started. Attach the screen capture that demo the results.
2. (2 mark) What Windows registry key should be modified to:
   1. Start **cmd** process automatically at boot time?

**HKEY\_CURRENT\_USER\Software\Microsoft\Windows\CurrentVersion\Run**

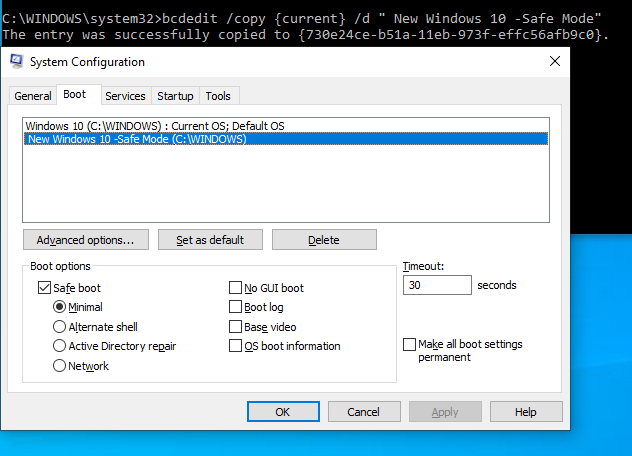
* 1. To control Ctrl-Alt\_DEL (CAD) ?

**HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\**

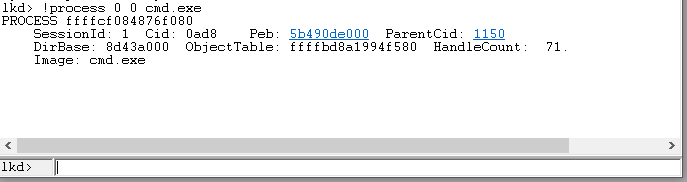
1. (2 marks) Attach a screen capture that demo how to use WMIC to display any system object (one of your choice) attributes.



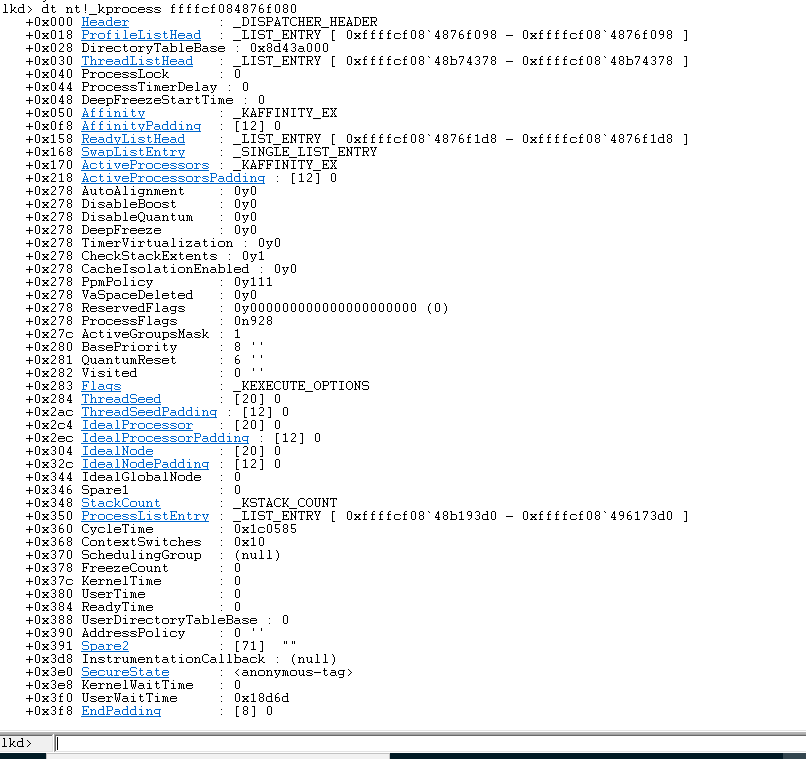
1. (2 marks) What Windows tool can be used to modify boot settings to start the system with minimal drivers. Attach screen capture that demo results



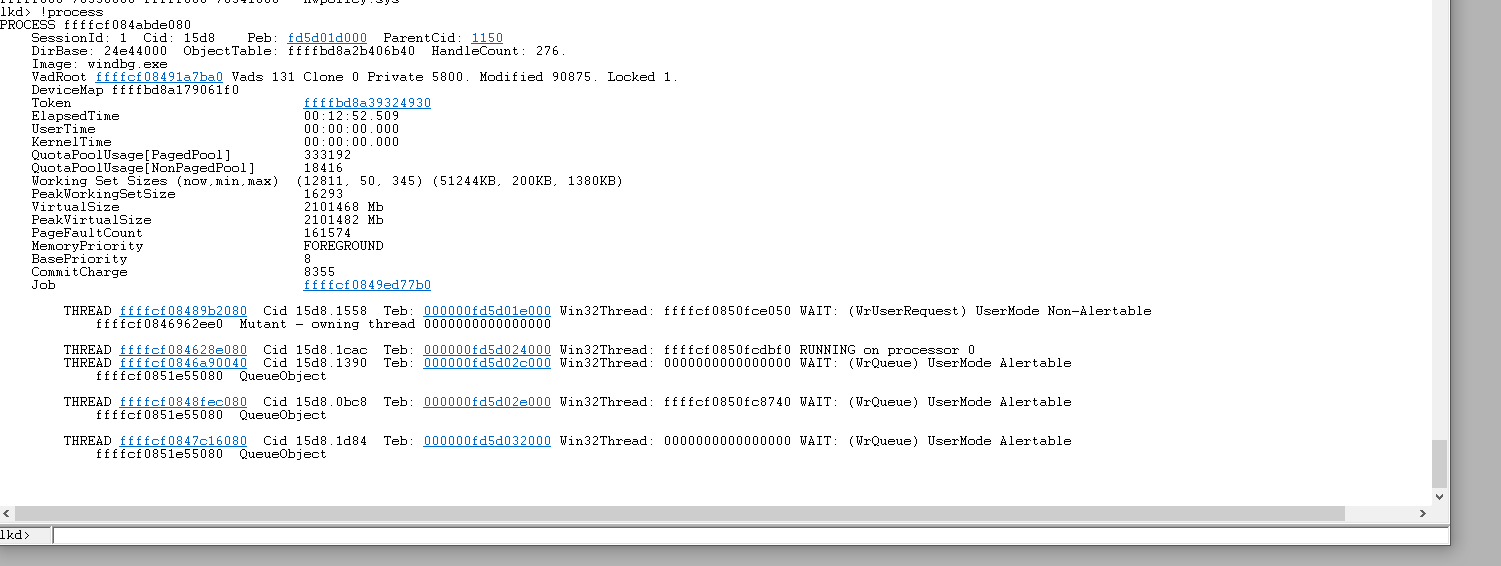
1. (4 marks) Start **cmd** process and use Windbg kernel commands to display the following for **cmd** process. Attach the respective screen captures to demo results.
   1. Process virtual address, pid and physical address of PDT (Page directory table)



* 1. PCB



* 1. Threads



* 1. peb

