

CS 236: Operating Systems Lab
Spring 2025-26
Labquiz3-Lite (20 marks)

A

Labquiz3-Lite

Proceed if you want to opt for the Lite version of CS236

Instructions

- No Internet. No phone. No devices. No handwritten notes. No books.
- No friends (strictly for the duration of the exam only!).
- Unethical practices will be an automatic recommendation for the FR grade.
- **Hardcoding of outputs will result in -5 marks per instance.**
- Please read the questions and submission instructions carefully.

Submission Guidelines

- **Untar the labquiz3-lite directory on your Desktop. Use:**
 - In *labquiz3-lite/q<no>*, *./run.sh* to run xv6 in containerized environment
 - In *labquiz3-lite*, *submit-xv6* to submit (ask a TA to enter the password)
- **Ensure that your final xv6 submission compiles without any errors. Submissions that do not compile will not be considered for evaluation.**

Some Important points

- **Add Test cases (file names) in Makefile after implementing the syscalls .**
- **You may need to change these files to implement a syscall**
 - **sysproc.c**
 - **syscall.h**
 - **syacall.c**
 - **user.h**
 - **usys.pl**

1. processes in action

(10 marks)

Write two system calls —

```
int getsyscalls(int pid);  
int validpagefaults(int pid);
```

The first system call should return the number of system calls made by the process with the mentioned **pid** and the second call should return the number of valid page faults due to memory accesses of the process with the mentioned **pid**.

Sample output

```
Shell  
$ p1a  
[3] num syscalls : 5
```

```
[3] num syscalls : 29

$ p1b
[3] num pagefaults : 0
[3] num pagefaults : 1
[3] num pagefaults : 2

$ p1c
[3] num pagefaults : 0
[3] num pagefaults : 1
[3] num pagefaults : 2
[3] num pagefaults : 2
[3] num pagefaults : 3

$ p1d
[3] num pagefaults : 0
[3] num pagefaults : 1
[3] num pagefaults : 2
[3] num pagefaults : 2
[3] num pagefaults : 3
[4] num pagefaults : 1
[4] num syscalls : 29
[3] num pagefaults : 3
[3] num syscalls : 165
```

2. unmapped virtual addresses (10 marks)

Add a system call **int getunmapped();**
that returns a count of the number of **valid** virtual address pages of a process with no physical page mappings.

Hint: **sbrklazy**

Sample output

```
Shell
$ p2a 1
```

```
Before allocation - Unmapped pages: 0
After allocation - Unmapped pages: 1
-----
Test passed: expected 1 unmapped pages, got 1
-----
```

```
Shell
$ p2a 5
Before allocation - Unmapped pages: 0
After allocation - Unmapped pages: 5
-----
Test passed: expected 5 unmapped pages, got 5
-----
```

```
Shell
$ p2b 5
Before allocation - Unmapped pages: 0
Allocating 5 pages with sbrklazy
After lazy allocation - Unmapped pages: 5
After touching 1 pages - Unmapped pages: 4
Again allocating 5 more pages with sbrklazy
After lazy allocation - Unmapped pages: 9
After touching 1 more page - Unmapped pages: 8
-----
Test passed: expected 8 unmapped pages, got 8
-----
```

```
Shell
$ p2c 5
Before allocation - Unmapped pages: 0
After lazy allocation - Unmapped pages: 5
After touching page 0 - Unmapped pages: 4
After touching page 1 - Unmapped pages: 3
After touching page 2 - Unmapped pages: 2
```

After touching page 3 - Unmapped pages: 1

After touching page 4 - Unmapped pages: 0

Test passed: expected 0 unmapped pages, got 0
