Department of Informatics and Telecommunications
Athens university
Operating Systems (K 22) / Period 2021-2022

2nd Work -

Copy Support during Recording (Copy On Write - COW) in xv 6

The problem - The existing implementation of the `fork () system call` in xv 6 copies all the memory (user - space) of the parent process to the child. In addition, this task is often useless, especially when the `fork ()` is followed by an `exec ()` in the child, which discards the copied memory without even having used most of it. However, if both processes (parent, child) use a page and one or both of them write on it, then a separate copy is required for each of them.

The solution - The purpose of copy support during recording (copy on write - cow) in `fork ()` is to postpone the assignment and copying of physical memory pages for the child until he / she really needs to use them if he / she really needs them.

`Fork ()` with copy support during registration creates only one page table for the child with the page table entries (Page Table Entries - PTE) to show on the parent's physical pages. The (user space) PTEs are declared non-registrable to both parent and child. When any of these processes try to write to one of these CoW pages, the CPU will cause a page error. The kernel page error handler will detect this case, assign a physical memory page to the process that caused the error, copy the homepage to the new one, and modify the relevant PTE in the process that caused the error, this time with the PTE as registrable. When the kernel page error handler returns, the user process will be able to modify (write) its own copy of the page.

CoW support for `fork () frees` user -assigned physical pages space memory more complex. A physical page can refer to multiple process page tables and should only be released when the last reference to it is removed.

Implement copying during recording (copy on write)

The work concerns the implementation of copying during the registration (copy on write) at kv xv 6. To test your implementation, `cowtest` and `usertests are provided` . To `cowtest` performs several tests, all of which fail in the original kernel format. First you will see:

```
$ cowtest
simple : fork () failed
$
```

The " simple " test assigns to a process more than half of the available physical memory and then calls the `fork ()` . H `fork ()` fails because there is not enough physical memory to give the child a complete copy of the parenting process memory.

When you have completed the task, the kernel should pass all the tests in the `cowtest` and `usertests programs` , ie:

```
$ cowtest
simple: ok
simple: ok
three: zombie!
ok
three: zombie!
ok
three: zombie!
ok
file: ok
ALL COW TESTS PASSED
$ usertests
...
ALL TESTS PASSED
$
```

Tips for implementing the work:

1. Modify the `uvmcopy ()` to match the parent's physical memory pages to the child, instead of assigning new pages. Reset the flag PTE _ W in the PTE of both parent and child.
2. Modify `usertrap ()` to detect page errors. When a page error occurs on a CoW page , assign a new page via `kalloc ()` , copy the old page to the new one, and install the new page in PTE with PTE _ W bit equal to 1.
3. PTE references to it are reset . You could keep a reference counter for each physical page, where the page tables listed on that page would be counted. When a page is initially assigned via `kalloc ()` , the report counter is initialized to 1. Increase the counter when `fork ()` creates a child sharing the page, and decrease the counter when a process removes the page from its page table. Kfree `( )` should return the page to the list of available pages only when the counter is reduced to 0. You can use an integer table to store counts and sizes equal to the total number of physical pages in the system.
4. Modify `copyout ()` to use the same mechanism as page errors when it encounters a CoW page .

Suggestions:

- PTE can be useful if it is a CoW mapping . You can use bits RSW (reserved for software) in RISC - V PTEs for this purpose.
- You will need the information in `kernel / riscv . h`
- If a CoW page error occurs and no memory is available, the process should be terminated.

Code

Use the code for the task as follows:

```
$ git clone git: //gallagher.di.uoa.gr/xv6-project-2021
Cloning into 'xv6-project-2021' ...
...
$ cd xv 6- project -2021
```

The repository xv 6- project -2021 adds a little extra functionality compared to the main one. To see the changes that have been made you can give the command

```
$ git log
```

The files you will need for this job are distributed through the git source code control system . You can find information about git in the git book or other public sources. Git allows you to keep track of all the changes you have made to the code . For example, if you finish a part of the job and want to record your changes locally, you can commit your changes using the command

```
$ git commit -am 'my solution for cow implementation project'
Created commit 60d2135: my solution for util lab exercise 1
1 files changed, 1 insertions (+), 0 deletions (-)
$
```

The date _ _ _ Delivery : 23 / Jan / 2021
Delivery method: submission to eclass , a tar file with all relevant files must be delivered .
Accompanying material: 3-4 page documentation explaining how you worked.
Implementation: the work is individual.
The work will be examined in the systems of the Division according to a program that will be announced after the delivery date.