

OS Assignment 2

1. Basics

- Firstly, the kernel.c code has been updated to include the frame buffer console. And boot.c has been modified to pass the kernel buffer as the first parameter.
-

Boot.c

- I have used AllocatePages() to ensure all the buffers are 4K aligned.
- The code in boot.c – function LoadFile() has been updated to include code for getting the file size of the kernel and user file, converting the size to pages and allocating a buffer using that page number.
- The user application is being loaded using the same steps as loading of the kernel.
- To reuse OpenFile() to read the paths for both user and kernel – I’ve initialized a new variable called type which takes 0 for kernel and 1 for user.
- So, if I pass 0 to my OpenFile() function, it means I have to read the kernel file from path - “[\\EFI\\BOOT\\KERNEL](#)”, otherwise if it’s 1 then read user file.
- The above technique of assigning 1 for user and 0 for kernel has been used for reusing functions to set up page tables as well in kernel.c.
- To take into account the kernel and user stack, I’ve allocated an extra page and shifted it by 4K bytes, ie. incremented the address by 0x1000 to point to the end of the stack.
- Parameters being passed to kernel:

Parameter Name	Meaning
kernel_addr	4K shifted (pointing to end of kernel stack) address to load the initial page table.
fb	Frame buffer base pointer
800	Width of fb
user_addr	4K shifted address (pointing to end of user stack and beginning of new page table) to load the new page table. 4 pages for the new page table + 1 page for user stack.
user_buffer	Physical address of the loaded user application. Pages = calculated from the size of the user file
user_pages	Number of pages to load user application. Used to set up page table entries for the user application. (Considering max size of 1Mb)

Kernel.c

- Set up the kernel page table first just as it was in Assignment 1.
- Next, added the user page table implementation.

- For the user page table – PTE[0] contains user stack, and remaining pages contain user application physical addresses.
- Level 4 – PML4[0] points to kernel's level 3 and PML4[511] points to user's level 3.
- Loaded the CR3 register with the new level 4.
- Calculated the virtual address which points to the top of the stack.
- User stack = Virtual address calculated above + 4K Bytes
- The user jump address is also same as user stack.

2. System Calls

User.c

- Placed two system calls using the __syscall1() function and cast the arguments to long.

KerneL_syscall.c

- Assigned the user stack pointer to the previously calculated user virtual address.
- Initialized syscall_entry_ptr to MSR_LSTAR register in syscall_init()
- Handled the system call by checking whether n=1 (for write) and printed the value.

OUTPUT

test [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

Framebuffer Console (ECE 6504)


Copyright (C) 2021 Ruslan Nikolaev

Allocated initial kernel page table.

Allocated new page table containing user.

System call 1

System call 2

 Right Ctrl