CSE231: Operating Systems

Assignment 2, Question 2

Name:	Mihir Chaturvedi
Roll no.:	2019061
Section:	B
Branch:	CSE

Addition of a System Call

We work on a cloned or extracted copy of the source of the Linux v5.9.1 kernel.

1. Add our new system call, with its identifying number, into the syscall table at `/usr/src/_linux-5.9.1/arch/x86/entry/syscalls/syscall_64.tbl`.
   ```
   440     common  sh_task_info            sys_sh_task_info
   ```
   This creates a reference to our system call.
2. In `/usr/src/_linux-5.9.1/kernel/sys.c`, we append our system-call function to the end of the file. This is the code that will run when we call our system-call from a user program.
3. Compile the modified kernel using `make`.
4. Install the modified kernel and its modules using `make modules_install install`.
5. Reboot the system to see the changes in effect.

The System Call

The function signature:

```
SYSCALL_DEFINE2(sh_task_info, long, pid, char *, us_path)
```

Here, our function is named *sh_task_info* and we accept two arguments: pid and path/filename. The function receives the task_struct corresponding to the PID provided and prints Process name, PID, state, priority, parent PID, virtual runtime and PIDs of all children processes into the kernel logs and into the file provided by the filename.

`kernel_write` has been used to write to the file, which is opened by `filp_open`. `printk` is used to log to the kernel.

User Inputs

1. **Process PID**
   This is the PID of the process of which we want the information of.
2. **Filename**
   This is the name or path (relative) of the file wherein we want to store the information of the requested process.

Expected Output

We will receive the output information for the request process in the specified file (say, “task_info.txt”). The following will get printed in the *task_info.txt* file and in the kernel logs:

```
Process details:
Name: systemd
PID: 1
State: 1
Priority: 120
Parent PID: 0
vruntime: 2384442366
Child 1 PID: 370
Child 2 PID: 493
Child 3 PID: 528
Child 4 PID: 569
Child 5 PID: 571
Child 6 PID: 609
Child 7 PID: 671
Child 8 PID: 678
Child 9 PID: 679
Child 10 PID: 681
Child 11 PID: 688
Child 12 PID: 689
Child 13 PID: 691
Child 14 PID: 694
Child 15 PID: 701
Child 16 PID: 703
Child 17 PID: 721
Child 18 PID: 726
Child 19 PID: 743
Child 20 PID: 744
Child 21 PID: 759
Child 22 PID: 859
Child 23 PID: 952
Child 24 PID: 1963
Child 25 PID: 1085
```

Error Values

1. **EINVAL:** This error will be returned when an invalid PID number has been passed. PID, being of positive `int` type, can only have values between 0 and 2147483647. Any other number provided will be treated as invalid.
2. **ESRCH:** This error will be returned when the requested PID does not correspond to any currently running process, and so the `task_struct` for the requested PID is NULL.
3. **EIO:** This error will be returned when there is some error in printing the text into the file.