

Scheduling In Linux

- > Implementing functions to custom schedule them according to the user.
- > Launching three threads, each of which relies on three different functions, countA(), countB() and countC().
- > Launching three processes, each of which relies on three different scripts to compile a custom linux kernel.
- > Uses clock_gettime() to get the current time.

Q1.1) Thread Scheduling

- > Using Linux's scheduling policies for three threads.
- > Launching three threads, each of which relies on three different functions, countA(), countB() and countC().
- > Each function does the same thing, i.e. counts from 1 – 2^32.
- > Uses clock_gettime() to get the current time.
- > Uses pthread_schedsetparam() to change priority of the threads.

Installation

```
# Navigate to src
$> cd src

# run the makefile
$> make -f MakeFile_Thread_1.mak
      OR
$> make
```

Results

Available at: /src/Plot_Thread_1.pdf

Q1.2) Process Scheduling

- > Using Linux's scheduling policies for three processes.
- > Launching three processes, each of which relies on three different scripts to compile a custom linux kernel.
- > Uses clock_gettime() to get the current time.
- > Uses sched_setscheduler() to change priority of the processes.

Installation

```
# Navigate to src
$> cd src

# run the makefile
$> make -f MakeFile_Process_1_2.mak
      OR
$> make
```

Results

Available at: /src/Plot_Process_1_2.pdf

Q2) Kernel Memory Copy

-> Creating a system call to read data bytes from user space and write back to user space .
-> In other words, this is a version of memcpy() that relies on the kernel to do the necessary copy operations, which are otherwise usually done directly in the user space. -> Uses clock_gettime() to get the current time.
-> Uses __copy_from_user() & __copy_to_user() to read bytes from user space and write back to user space.

Installation

```
#Navigate to src
$> cd src

# Compile the Kernel located in /res
$> ./t

# run the diff
$> diff --normal /linux-5.19.8 /linux-5.19.8_Norm
```

diff

```
2789a2790,2799
> SYSCALL_DEFINE4(kernel_2d_memcpy,float *, src,float *,dest,int, row,int,col){
>     float buffer[row][col];
>     if(__copy_from_user(buffer,src,sizeof(int)* (buffer))){
>         return -EFAULT;
>     }
>     if(__copy_to_user(dest,buffer,sizeof(int)* (buffer))){
>         return -EFAULT;
>     }
>     return 0;
> }
```

Results

Available at: /src/DIFF_SYSCALLDEFINE_2.pdf

License

MIT © Arnav Gupta 2022
Original Creator - [Arnav Gupta](#)

MIT License

Copyright (c) 2022 Arnav Gupta

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

--- EOF ---