

Modified xv6-riscv

Specifications:

1. Syscall Tracing

- Created user/strace.c file which is used to call trace system calls.
- Added trace system call in kernel/syscall.h
- In the Makefile, \$U/_strace was appended to the UPROGS
- Addition related to trace system call is done to the kernel/syscall.c as well as in syscall() function to enable the printing of trace output.
- Created sys_trace() function to create a new system call for setting the trace mask.
- To enable the integration of the new system call, a placeholder has been added to the user/usys.pl script, and a corresponding declaration for the system call has been included in the user/user.h header file.
- In Process of tracing, It should accept a single argument, which is an integer mask. The set bit of mask value determines which system calls should be traced.

List of files where changes are made:

1. Makefile.mk
2. kernel/proc.c
3. kernel/proc.h
4. kernel/sysproc.c
5. kernel/syscall.h
6. user/strace.c
7. user/usys.S
8. user/usys.pl

Execution:

Demonstrate the use of the “strace” program by running it with a sample command. Here’s an illustration using the “grep” command to search for the term “hello” in a file called “README”.

Bash Command :

strace mask [command] [args]

```
$ ls
.          1  1  1024
..         1  1  1024
README    2  2  2225
cat        2  3  23872
echo       2  4  22712
forktest   2  5  13440
grep       2  6  27016
init       2  7  23520
kill       2  8  22632
ln         2  9  22496
ls         2 10  26056
mkdir      2 11  22768
rm         2 12  22752
sh         2 13  40768
stressfs   2 14  23720
time       2 15  23024
usertests  2 16 150464
grind      2 17  37256
wc         2 18  24848
zombie     2 19  22008
schedulertest 2 20 23408
strace     2 21  22920
setpriority 2 22 23328
console    3 23   0
$
```

```
xv6 kernel is booting
hart 2 starting
hart 1 starting
init: starting sh
$ strace 32 grep hello README
3: syscall read (3 2808 1023) -> 1023
3: syscall read (3 2863 968) -> 968
3: syscall read (3 2813 1018) -> 234
3: syscall read (3 2845 986) -> 0
$
```

2. Scheduling

When no specific user preferences are provided, a predefined scheduling algorithm (Round Robin) is set as the default to ensure the kernel functions seamlessly.

a) FCFS(First Come First Serve)

- FCFS prioritizes the process with the earliest creation time, which is determined by the tick number associated with when the process was generated. This chosen process remains in execution until it reaches termination.
- The only change made was in the `kernel/proc.c` file. We added a loop to find the process with the earliest creation time, which is represented by `struct proc::ctime`. This value corresponds to the number of ticks at which the process was allocated and initialized.
- To selectively disable preemption, we modified the conditional use of the `yield()` function in both `usertrap()` and `kerneltrap()` within the

kernel/trap.c file. This modification depends on the chosen scheduling method. In the case of FCFS scheduling, preemption has been deactivated.

Execution:

```
aconoti0@aconoti0:~/Desktop/enhanced-xv6-riscv-riscv$ make qemu SCHEDULER=FCFS
qemu-system-riscv64 -machine virt -bios none -kernel kernel/kernel -n 128M -smp 3 -nographic -drive file=fs.img,if=none,format=raw,id=x0 -device virtio-blk-device,drive=x0,bus=virtio-mmio-bus.0

xv6 kernel is booting

hart 1 starting
hart 2 starting
init: starting sh
$
PID  Prio  State  rtime  wtime  nrun  q0    q1    q2    q3    q4
1    0     sleep  0      35     10    0     0     0     0     0
2    0     sleep  0      34     7     0     0     0     0     0
```

b) PBS(Priority Based Scheduling)

- PBS (Priority-Based Scheduling) is a non-preemptive scheduler that chooses the process with the highest priority to execute.
- To support PBS, the some variables were added to the 'struct proc' in 'kernel/proc.h' like rtime(running time),stime(sleep time),no_of_times_scheduled
- Default values for these variables were set during their initialization in the 'allocproc()' function found in the 'kernel/proc.c' file.
- We introduced scheduling functionality for PBS (Priority-Based Scheduling), which computes the dynamic priority of processes by considering both their static priority and the 'niceness' value.
- Niceness has the following impact on Dynamic Priority (DP):
 - (a) A niceness value of 5 has no effect on DP; it is considered neutral.
 - (b) A niceness value of 10 increases priority by 5.
 - (c) A niceness value of 0 decreases priority by 5.

$\text{niceness} = \text{INT}(\text{sleep}/(\text{sleep} + \text{Running}) * 10)$

$D_p = \max(0, \min(\text{SP} - \text{niceness} + 5, 100))$

- A 'set_priority()' function was added in 'kernel/proc.c' to facilitate the adjustment of process priorities.
- A user program called 'user/setpriority.c' was developed to enable users to configure the priorities of processes.
- To interface with the 'set_priority()' function, we introduced a 'sys_set_priority()' system call within 'kernel/sysproc.c.' This system call empowers users to make adjustments to process priorities as required.

```
xv6 kernel is booting
hart 2 starting
hart 1 starting
init: starting sh
$
PID      Prio    State   rtime   wtime   nrun
1         60     sleep   0        118     21
2         60     sleep   0        117     7
```

c) PBS(Priority Based Scheduling)

For priority 0 : 1 timer tick

For priority 1 : 2 timer ticks

For priority 2 : 4 timer ticks

For priority 3 : 8 timer ticks

For priority 4 : 16 timer ticks

- In the Multi-Level Feedback Queue (MLFQ) implementation, we have incorporated an aging mechanism to prevent processes from getting starved. This is achieved by using a parameter called `WAITING_LIMIT`. Here's how it works:
- If the difference between the current 'ticks' and the 'EntryTime' of a process exceeds the 'waiting_limit', then:
- The process is moved up to a higher-priority queue.
- The 'EntryTime' of the process is updated in the new queue.
- The queue number of the process is reduced, effectively increasing its priority.
- Declare a `currProc` pointer of `proc` type and initialize it to 0. Set the highest queue value to 5.
- Iterate through the `PROC` array in a loop. While doing so, locate the runnable process. Once identified, choose the process that belongs to the lowest-priority queue and assign it as the selected process. Additionally, mark the corresponding queue as the highest-priority queue. In case two processes are in the same queue with the highest priority, resolve the tie by selecting the one with the lower 'p->entrytime.'

Bash Command :

`make clean`

`make qemu SCHEDULER=XXXX [DEFAULT | FCFS | PBS | MLFQ]`

```
xv6 kernel is booting
```

```
hart 1 starting
```

```
hart 2 starting
```

```
init: starting sh
```

```
$
```

PID	Prio	State	rtime	wtime	nrun	q0	q1	q2	q3	q4
1	0	sleep	0	13	21	0	0	0	0	0
2	0	sleep	0	12	7	0	0	0	0	0

3. Procdump function enhancement

List of files where changes are made:

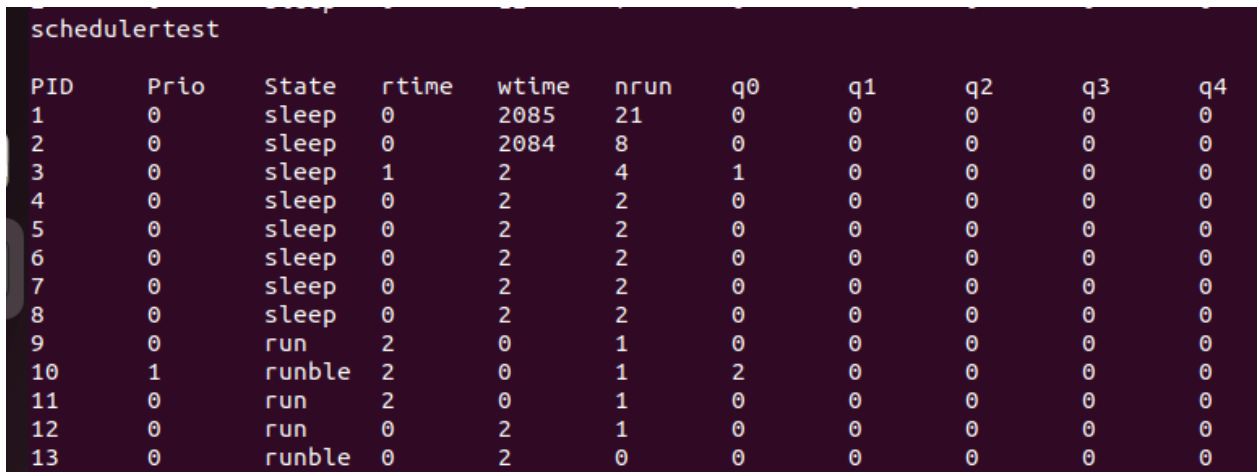
- 1.Makefile.mk
- 2.kernel/proc.c
- 3.kernel/proc.h

- This function is valuable for debugging the scheduling algorithms we've implemented in step 2.
- It provides insights into various parameters that were calculated during the scheduling implementation process, and these parameters have been included in the procdump output for analysis and debugging purposes.

Bash Command :

schedulertest

Ctrl + P



PID	Prio	State	rtime	wtime	nrun	q0	q1	q2	q3	q4
1	0	sleep	0	2085	21	0	0	0	0	0
2	0	sleep	0	2084	8	0	0	0	0	0
3	0	sleep	1	2	4	1	0	0	0	0
4	0	sleep	0	2	2	0	0	0	0	0
5	0	sleep	0	2	2	0	0	0	0	0
6	0	sleep	0	2	2	0	0	0	0	0
7	0	sleep	0	2	2	0	0	0	0	0
8	0	sleep	0	2	2	0	0	0	0	0
9	0	run	2	0	1	0	0	0	0	0
10	1	runble	2	0	1	2	0	0	0	0
11	0	run	2	0	1	0	0	0	0	0
12	0	run	0	2	1	0	0	0	0	0
13	0	runble	0	2	0	0	0	0	0	0

4. Benchmark Testing

A new file created in the `user/schedulertest.c` was added to test the implemented schedulers.

- **Round Robin**

Average run time: 106

Average waiting time 10

- **First Come First Serve**

Average run time: 25

Average waiting time 20

- **Priority Based Scheduling**

Average run time: 104

Average waiting time 10

- **Multilevel feedback queue**

Average run time: 106

Average waiting time : 9