LAB #2: MODIFYING XV6 SCHEDULER

Group Members:

- Yash Aggarwal (yagga004)
- Nityash Gautam (ngaut006)
- Parth Bhatt (pbhat029)

Demonstration Link:

Click Here

PART 1: SYSTEM CALLS

Similar to the previous Lab Assignment, System Calls have been added.

1. 2 new syscall numbers have been defined at **kernel/syscall.h**
2. The syscall table at **kernel/syscall.c** has been updated with respect to the new syscalls added.
3. A syscall function "*sys_sched_statistics*", has been defined at **kernel/sysproc.c**.
    a. *sched_statistics* iterates all the processes in the process table, and prints the results as per the UNUSED processes.
4. "*sys_sched_statistics*" is the return to the kernel function "*sched_statistics()*".
5. The argument passed into the "*sys_sched_ticekts*" is the ticket assigned to the process, and is adjusted to the range (0, MAX_NUM_TICKET_PER_PROCESS).
6. After this, the kernel function "*sched_tickets*" is called.
7. "*sched_tickets(tickets)*" sets the caller's process tickets to the desired value.
8. Further, inside the *scheduler()*, if the process is chosen, the number of ticks is incremented.
9. The function is defined at **kernel/defs.h**
10. The user space is updated at **user/usys.pl** and **usys/user.h**
11. The **"*$U/_lab2_*\"** is added to the **MakeFile** and the CPU is changed to 1.

PART 2: SCHEDULER IMPLEMENTATION

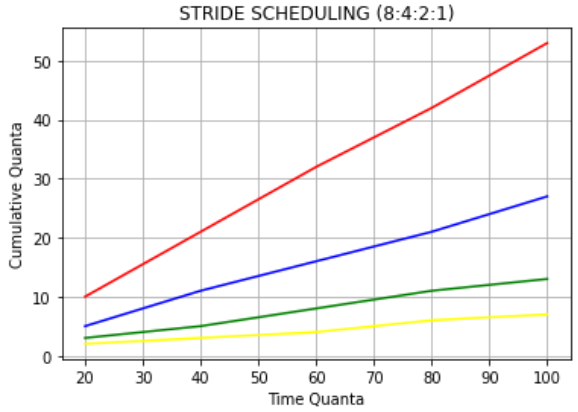
The Scheduler Implementation has been done following the below-mentioned steps:

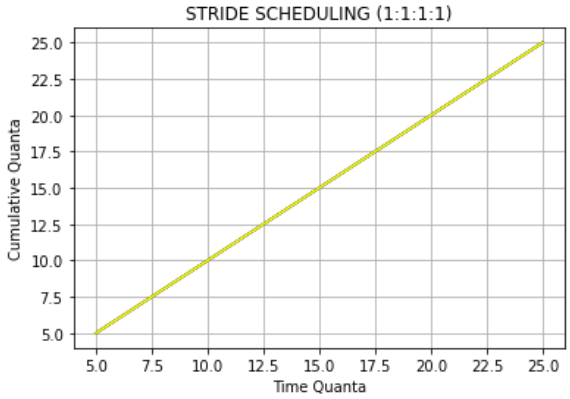
1. 2 Variables (strides & pass) are added in the *struct proc* at the **kernel/proc.h** file.
2. At the **kernel/proc.c**, the "*rand() function*" is added as per the Lab2 Instructions provided.
3. Inside the "*allocproc()*" function in the **kernel/proc.c** file, a new variable of struct proc is initialized before the process is returned.
4. In the **kernel/proc.c** file, inside the scheduler function, we use #if define…#end per instruction
    a. Inside the infinite Loop, the original second loop is moved to the #else branch.
    b. Within the "#if defined(LOTTERY)" branch:
        i. The process table is iterated to find the runnable processes.

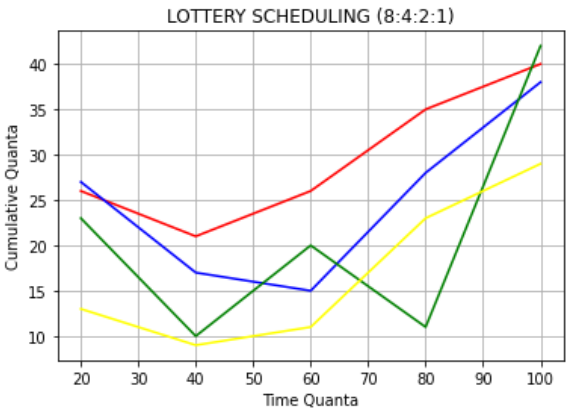
ii. Total tickets of the Runnable processes are calculated.
iii. Finally, a random number is generated within the range.
iv. Also, the total number of tickets till this point "*tickets_sum*" is calculated.
v. If "*tickets_sum*" is less than the LOTTERY, the next runnable process is executed. Also, the number of ticks is increased, as and when the runnable process is selected.

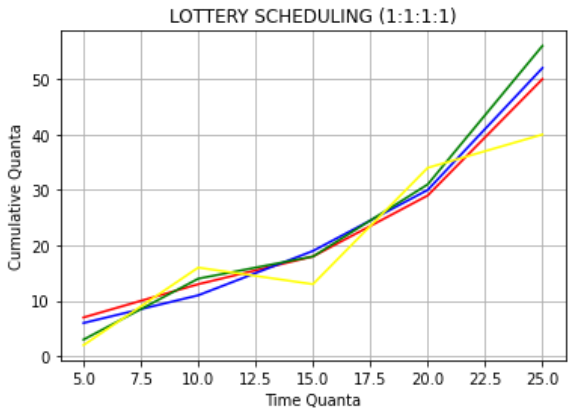
c. Within the "#if defined(STRIDE)" branch:
i. The process table is iterated to find the process with the minimum pass.
ii. Once found, the process's pass value is incremented by strides (K/#ticket, at p->strides)
iii. Once done, we switch and increase ticks as normal.

PART 3: EXPERIMENTS









From the results obtained above, it can easily be observed that the LOTTERY SCHEDULING algorithm shows significant variability on the given time scales.

This is the result of the fact that Lottery Scheduling uses a Probabilistic technique for ticket scheduling.

Observing the deterministic nature of the stride scheduling algorithm, we observe the results produced by it are periodic in nature.

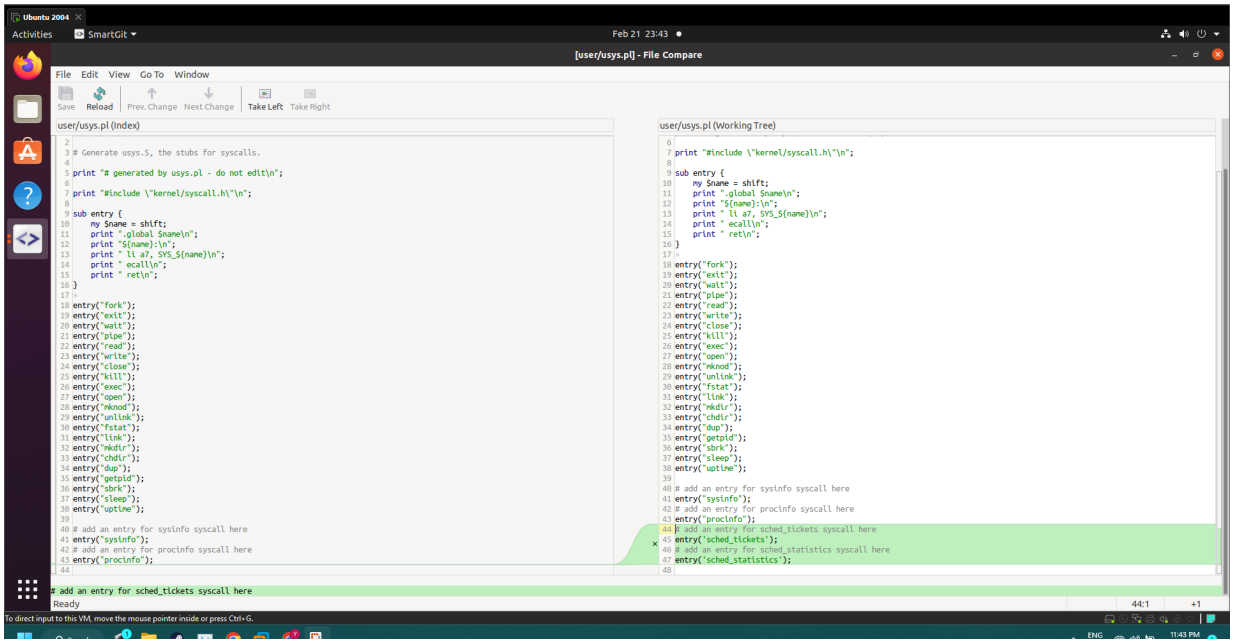
CONCLUSION: The stride Scheduling Algorithm is more suitable than Lottery Scheduling, because of its deterministic nature of scheduling strides.

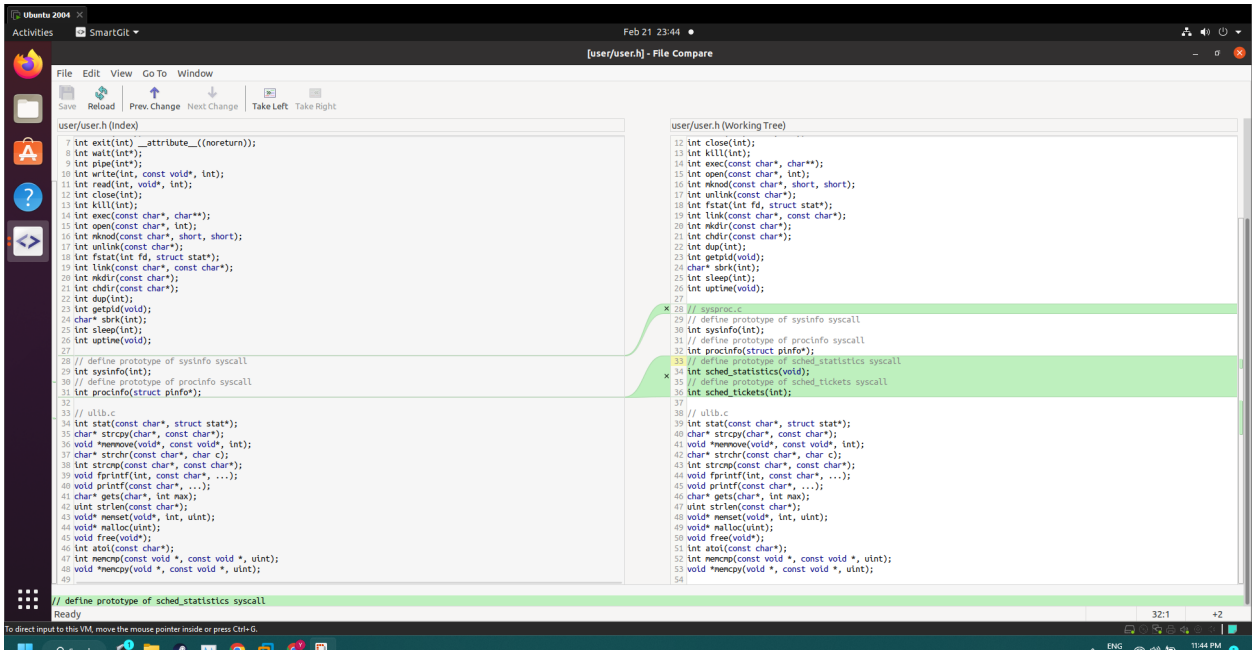
CONTRIBUTIONS

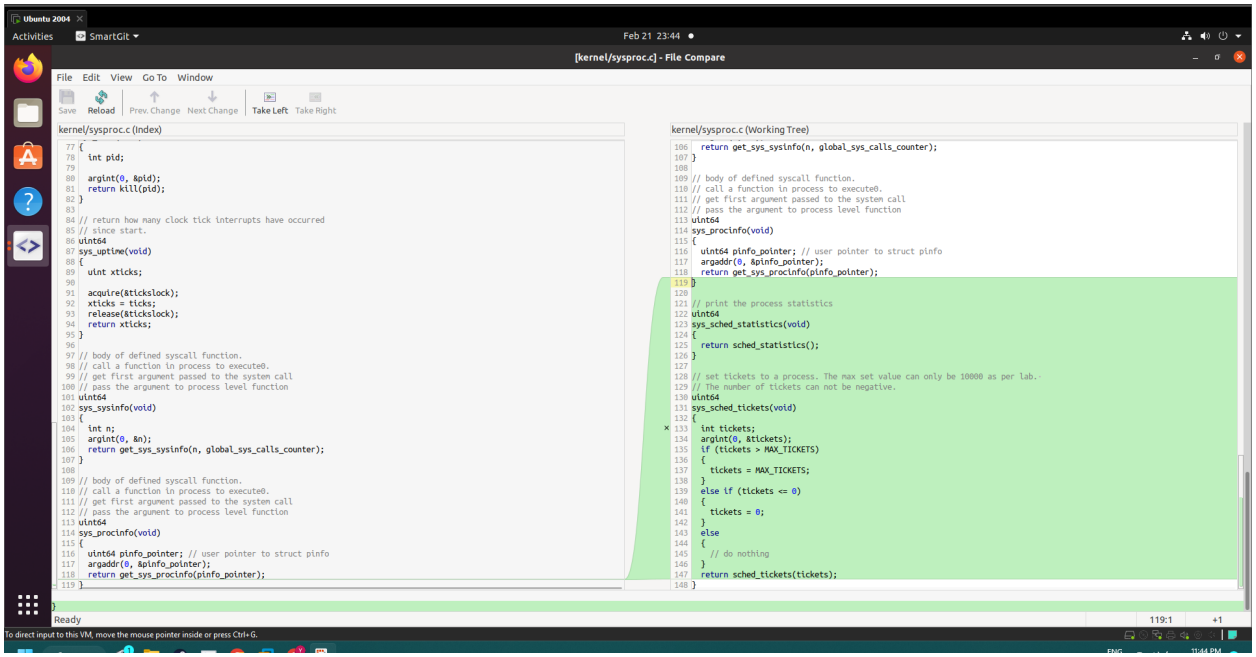
Yash Aggarwal: Design and implementation of Lottery Scheduling.
Parth Bhatt: Design and implementation of Stride Scheduling.
Nityash Gautam: Implementing the System Calls and Experimentation.

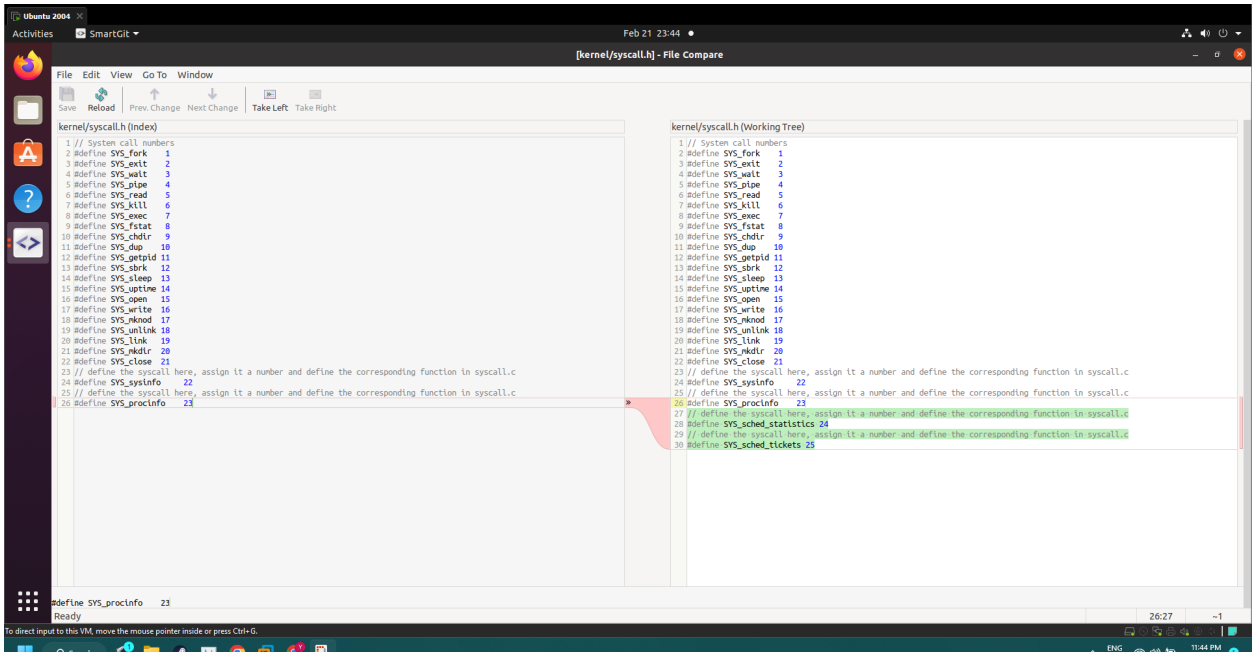
Note*: The above-mentioned Contributions are not strict. Every member contributed to the working implementation of the mentioned sections of the Lab Assignments.

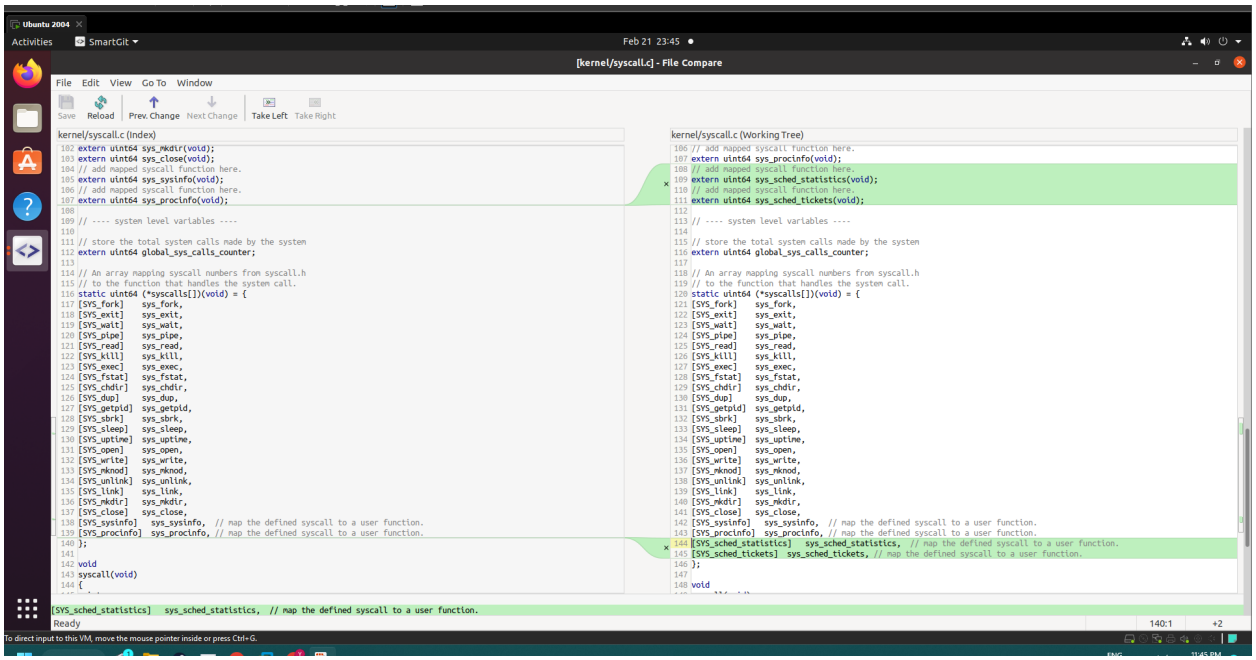
CODE CHANGES

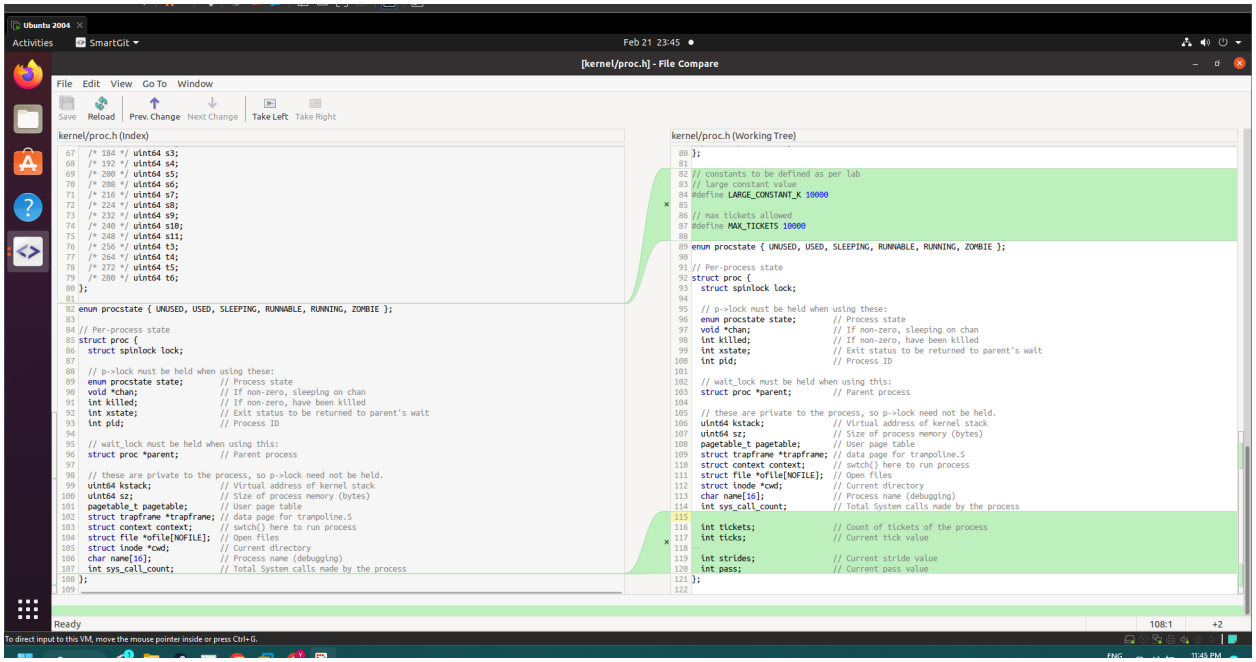


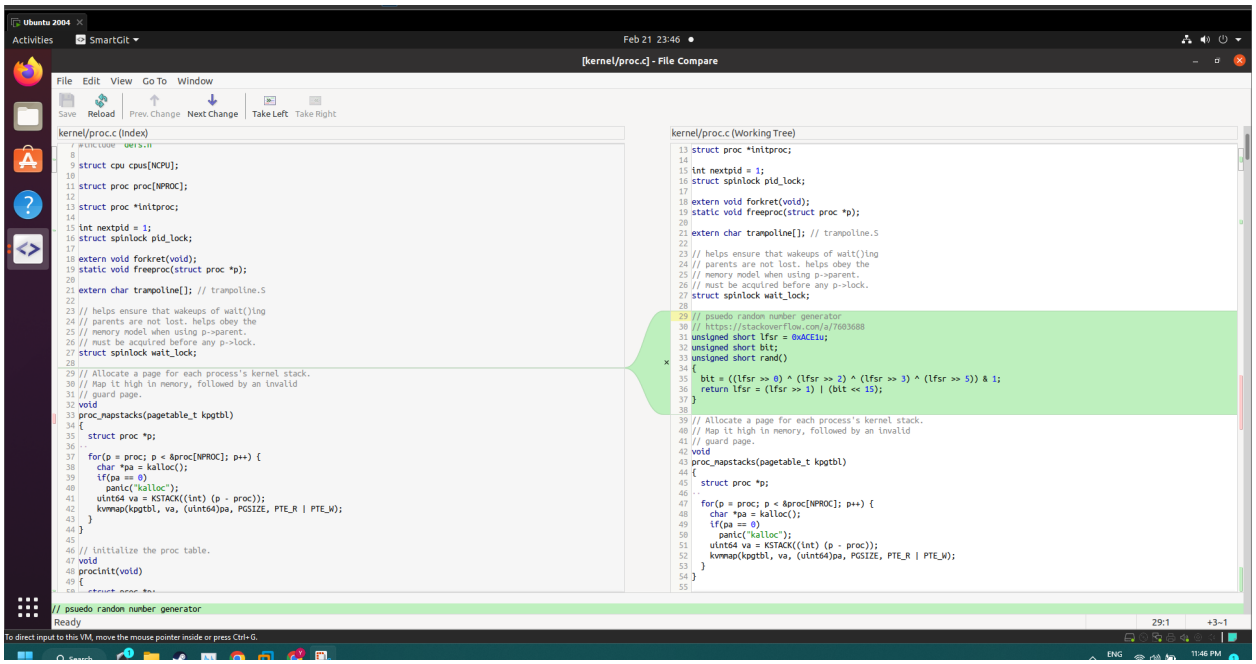

Ubuntu 2004
Activities
SmartGit
Feb 21 23:44
[user/user.h] - File Compare
File Edit View Go To Window
Save Reload Prev. Change Next Change Take Left Take Right
user/user.h (Index)
user/user.h (Working Tree)
// define prototype of sysinfo syscall
int sysinfo(int);
// define prototype of procinfo syscall
int procinfo(struct pinfo*);
// define prototype of sched_statistics syscall
int sched_statistics(void);
// define prototype of sched_tickets syscall
int sched_tickets(int);
// define prototype of sched_statistics syscall
Ready
32:1
+2
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

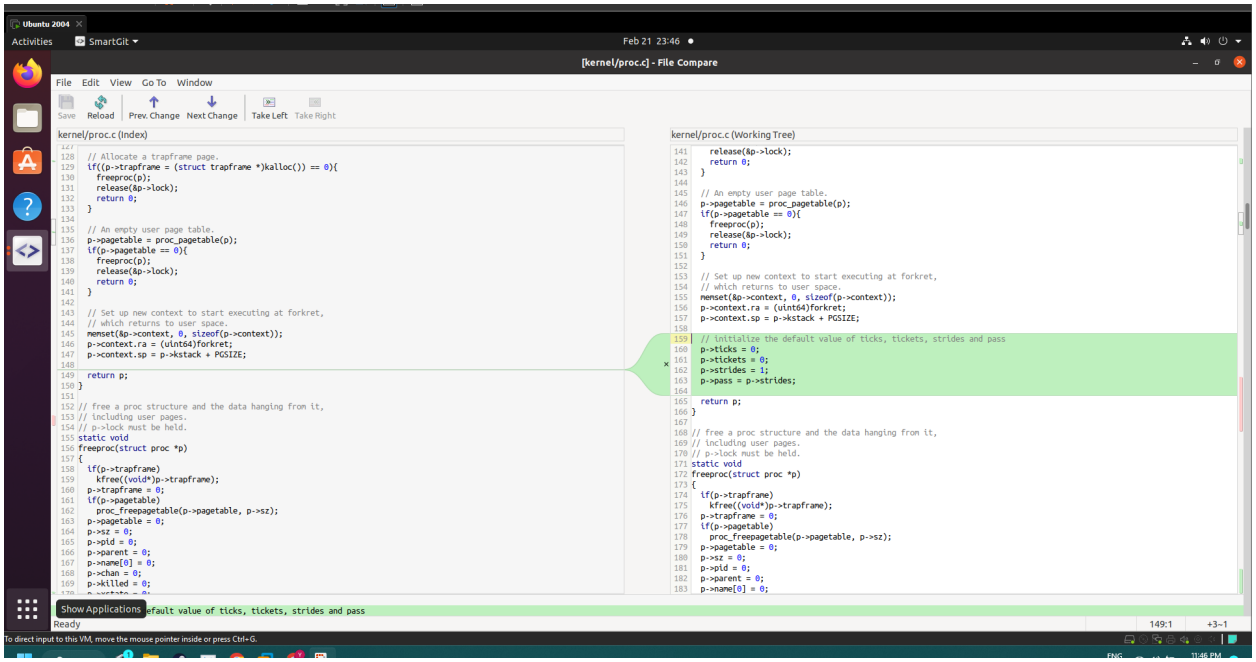

Ubuntu 2004
Activities
SmartGit
Feb 21 23:44
[kernel/sysproc.c] - File Compare
File Edit View Go To Window
Save Reload Prev. Change Next Change Take Left Take Right
kernel/sysproc.c (Index)
kernel/sysproc.c (Working Tree)
// return how many clock tick interrupts have occurred
// since start.
// body of defined syscall function.
// call a function in process to execute0.
// get first argument passed to the system call
// pass the argument to process level function
// print the process statistics
// set tickets to a process. The max set value can only be 10000 as per lab.
// The number of tickets can not be negative.
Ready
119:1
+1
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

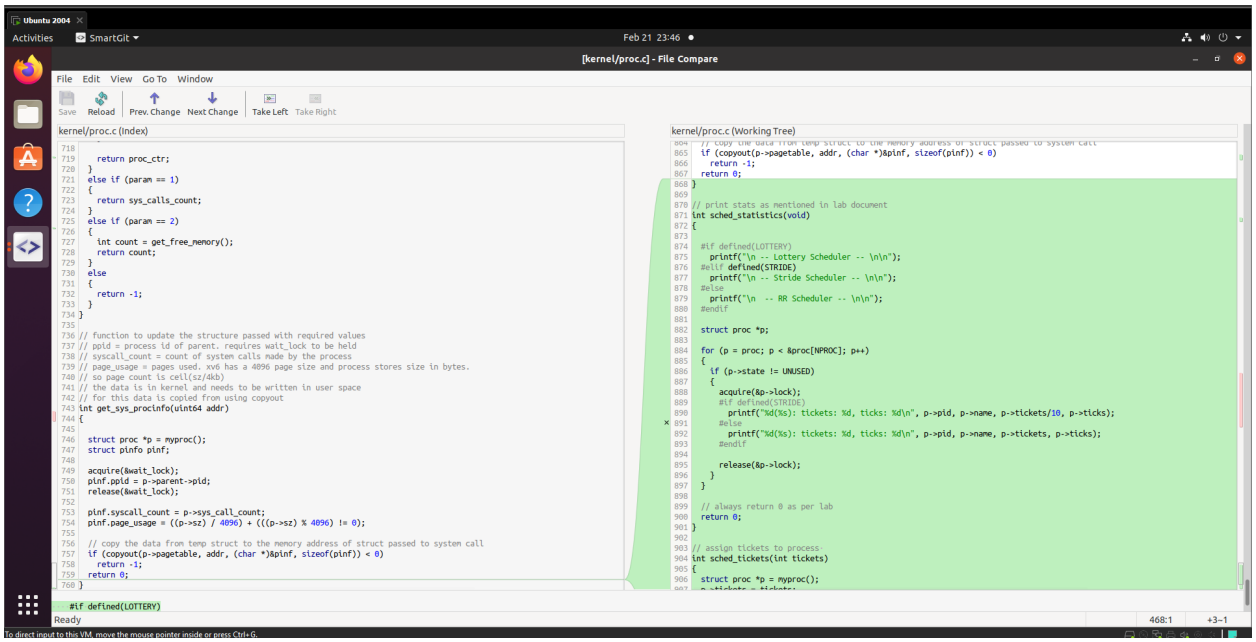

Ubuntu 2004
Activities
SmartGit
Feb 21 23:44
[kernel/syscall.h] - File Compare
File Edit View Go To Window
Save Reload Prev. Change Next Change Take Left Take Right
kernel/syscall.h (Index)
kernel/syscall.h (Working Tree)
// System call numbers
#define SYS_fork 1
#define SYS_exit 2
#define SYS_wait 3
#define SYS_pipe 4
#define SYS_read 5
#define SYS_kill 6
#define SYS_exec 7
#define SYS_fstat 8
#define SYS_chdir 9
#define SYS_dup 10
#define SYS_getpid 11
#define SYS_sbrk 12
#define SYS_sleep 13
#define SYS_uptime 14
#define SYS_open 15
#define SYS_write 16
#define SYS_mknod 17
#define SYS_unlink 18
#define SYS_link 19
#define SYS_mkdir 20
#define SYS_close 21
// define the syscall here, assign it a number and define the corresponding function in syscall.c
#define SYS_sysinfo 22
// define the syscall here, assign it a number and define the corresponding function in syscall.c
#define SYS_procinfo 23
// define the syscall here, assign it a number and define the corresponding function in syscall.c
#define SYS_sched_statistics 24
// define the syscall here, assign it a number and define the corresponding function in syscall.c
#define SYS_sched_tickets 25
#define SYS_procinfo 23
Ready
26:27
~1
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
ENG
11:44 PM

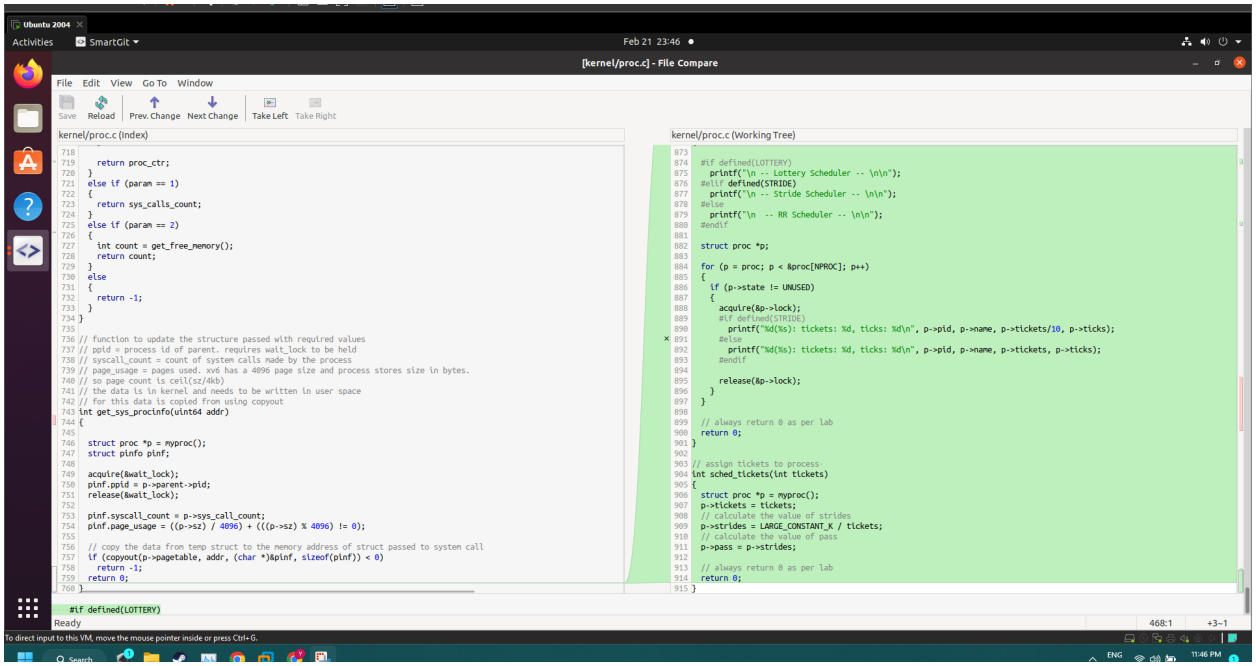

Ubuntu 2004
Activities
SmartGit
Feb 21 23:45
[kernel/syscall.c] - File Compare
File Edit View Go To Window
Save Reload Prev. Change Next Change Take Left Take Right
kernel/syscall.c (Index)
kernel/syscall.c (Working Tree)
extern uint64 sys_mkdir(void);
extern uint64 sys_close(void);
// add mapped syscall function here.
extern uint64 sys_sysinfo(void);
// add mapped syscall function here.
extern uint64 sys_procinfo(void);
// add mapped syscall function here.
extern uint64 sys_sched_statistics(void);
// add mapped syscall function here.
extern uint64 sys_sched_tickets(void);
// ---- system level variables ----
// store the total system calls made by the system
extern uint64 global_sys_calls_counter;
// An array mapping syscall numbers from syscall.h
// to the function that handles the system call.
static uint64 (*syscalls[])(void) = {
[SYS_fork] sys_fork,
[SYS_exit] sys_exit,
[SYS_wait] sys_wait,
[SYS_pipe] sys_pipe,
[SYS_read] sys_read,
[SYS_kill] sys_kill,
[SYS_exec] sys_exec,
[SYS_fstat] sys_fstat,
[SYS_chdir] sys_chdir,
[SYS_dup] sys_dup,
[SYS_getpid] sys_getpid,
[SYS_sbrk] sys_sbrk,
[SYS_sleep] sys_sleep,
[SYS_uptime] sys_uptime,
[SYS_open] sys_open,
[SYS_write] sys_write,
[SYS_mknod] sys_mknod,
[SYS_unlink] sys_unlink,
[SYS_link] sys_link,
[SYS_mkdir] sys_mkdir,
[SYS_close] sys_close,
[SYS_sysinfo] sys_sysinfo, // map the defined syscall to a user function.
[SYS_procinfo] sys_procinfo, // map the defined syscall to a user function.
[SYS_sched_statistics] sys_sched_statistics, // map the defined syscall to a user function.
[SYS_sched_tickets] sys_sched_tickets, // map the defined syscall to a user function.
};
void
syscall(void)
{
[SYS_sched_statistics] sys_sched_statistics, // map the defined syscall to a user function.
Ready
140:1
+2
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

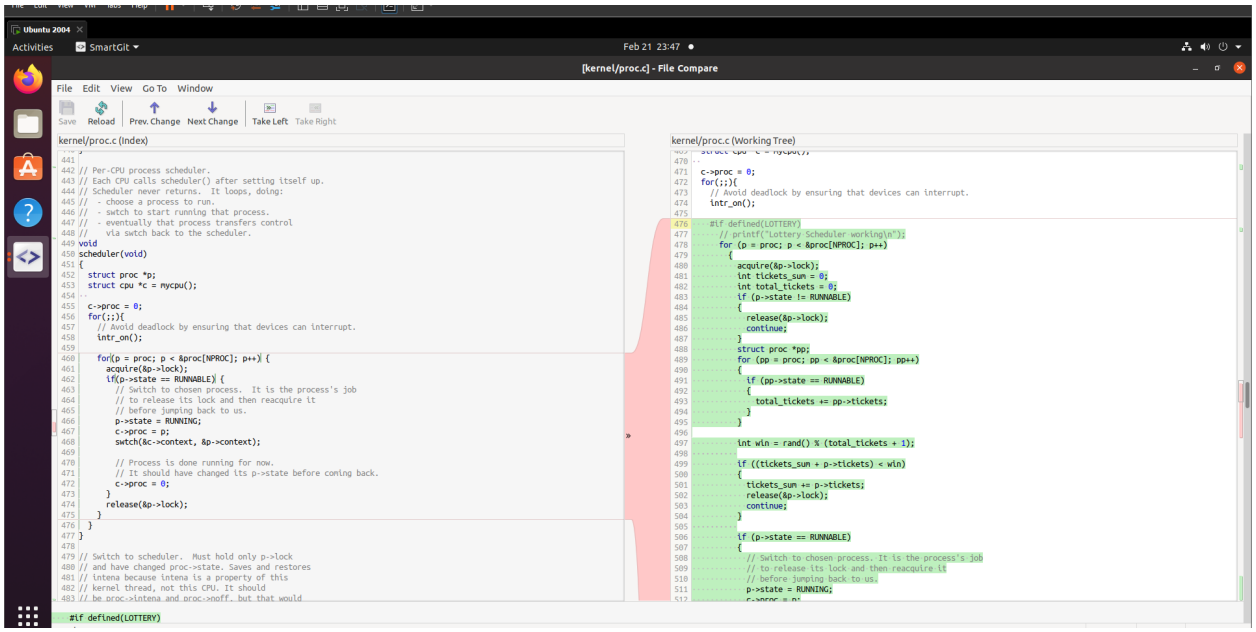

Ubuntu 2004
Activities
SmartGit
Feb 21 23:45
[kernel/proc.h] - File Compare
File Edit View Go To Window
Save Reload Prev. Change Next Change Take Left Take Right
kernel/proc.h (Index)
kernel/proc.h (Working Tree)
Ready
108:1
+2
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

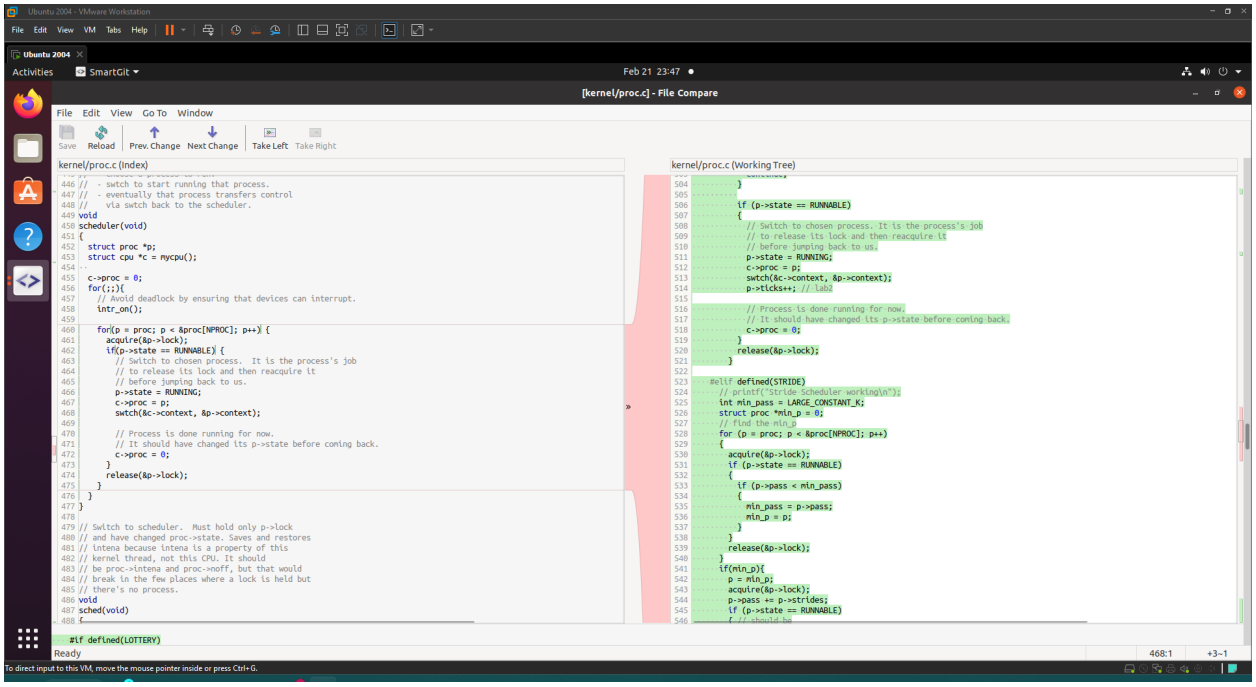

Ubuntu 2004
Activities
SmartGit
Feb 21 23:46
[kernel/proc.c] - File Compare
File Edit View Go To Window
Save Reload Prev. Change Next Change Take Left Take Right
kernel/proc.c (Index)
kernel/proc.c (Working Tree)
// psuedo random number generator
Ready
29:1
+3~1
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

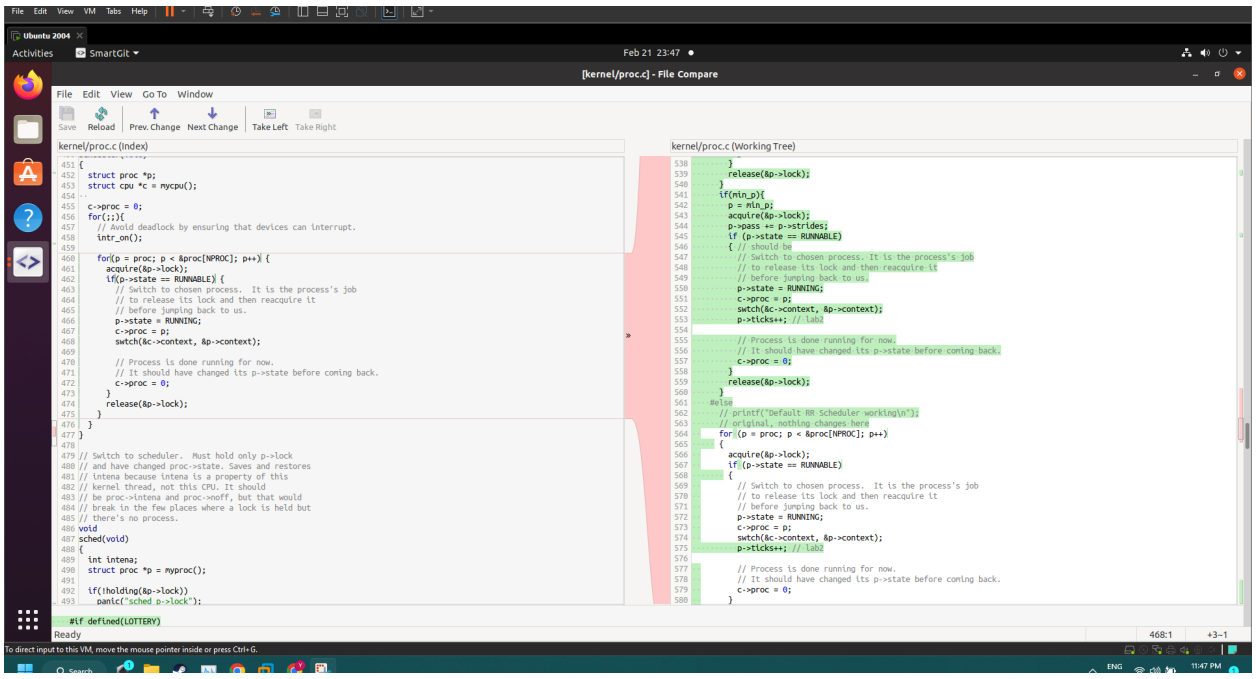

Ubuntu 2004
Activities
SmartGit
Feb 21 23:46
[kernel/proc.c] - File Compare
File Edit View Go To Window
Save Reload Prev. Change Next Change Take Left Take Right
kernel/proc.c (Index)
kernel/proc.c (Working Tree)
Show Applications
Ready
149:1 +3~1
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

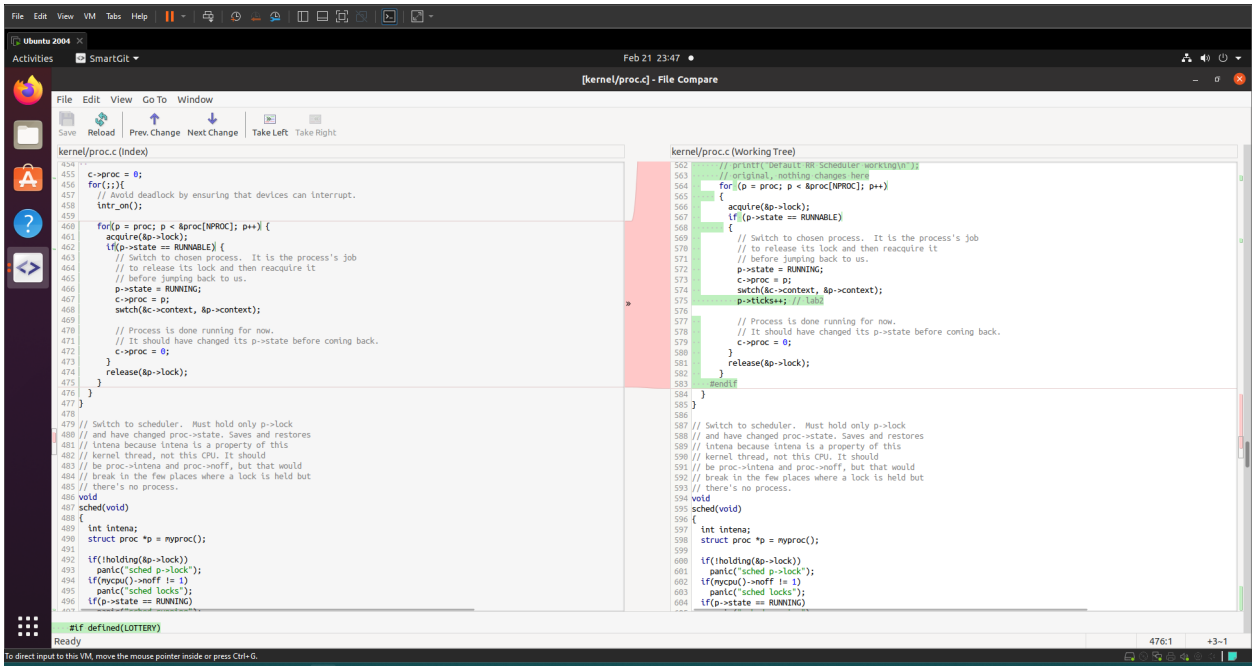

Ubuntu 2004
Activities
SmartGit
Feb 21 23:46
[kernel/proc.c] - File Compare
File Edit View Go To Window
Save Reload Prev. Change Next Change Take Left Take Right
kernel/proc.c (Index)
kernel/proc.c (Working Tree)
Ready
468:1 +3~1
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

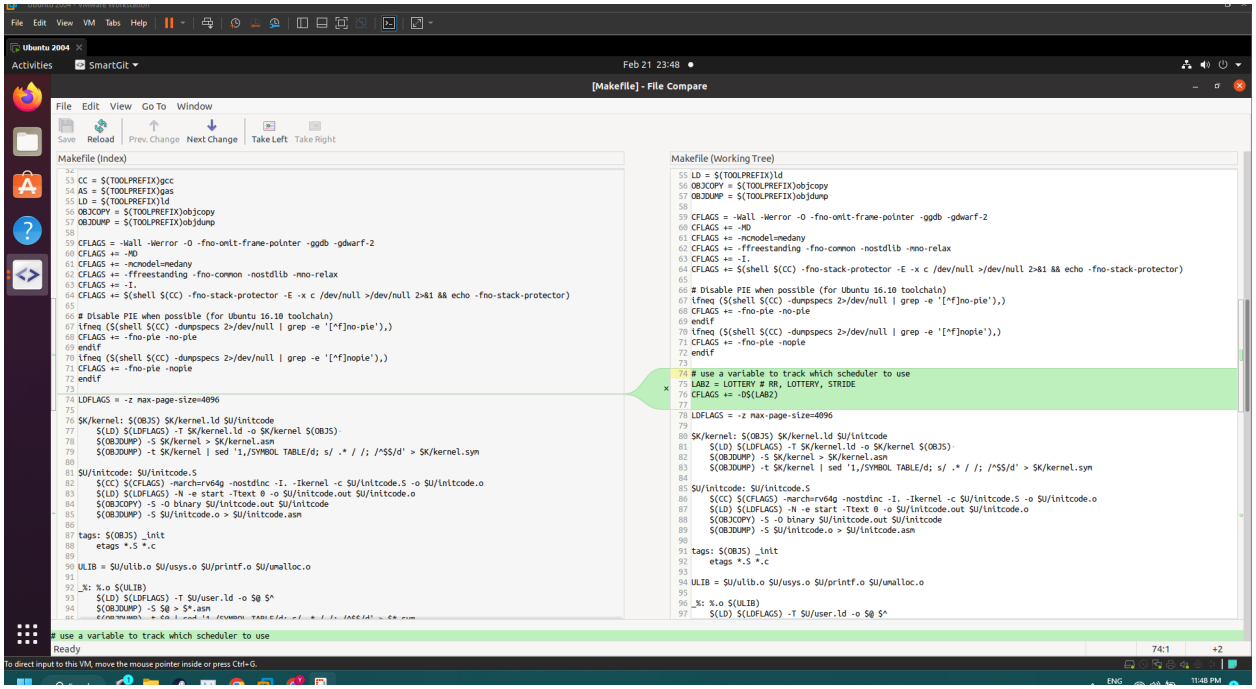

Ubuntu 2004
Activities
SmartGit
Feb 21 23:46
[kernel/proc.c] - File Compare
File Edit View Go To Window
Save Reload Prev. Change Next Change Take Left Take Right
kernel/proc.c (Index)
kernel/proc.c (Working Tree)
#if defined(LOTTERY)
Ready
468:1
+3~1
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
ENG
11:46 PM

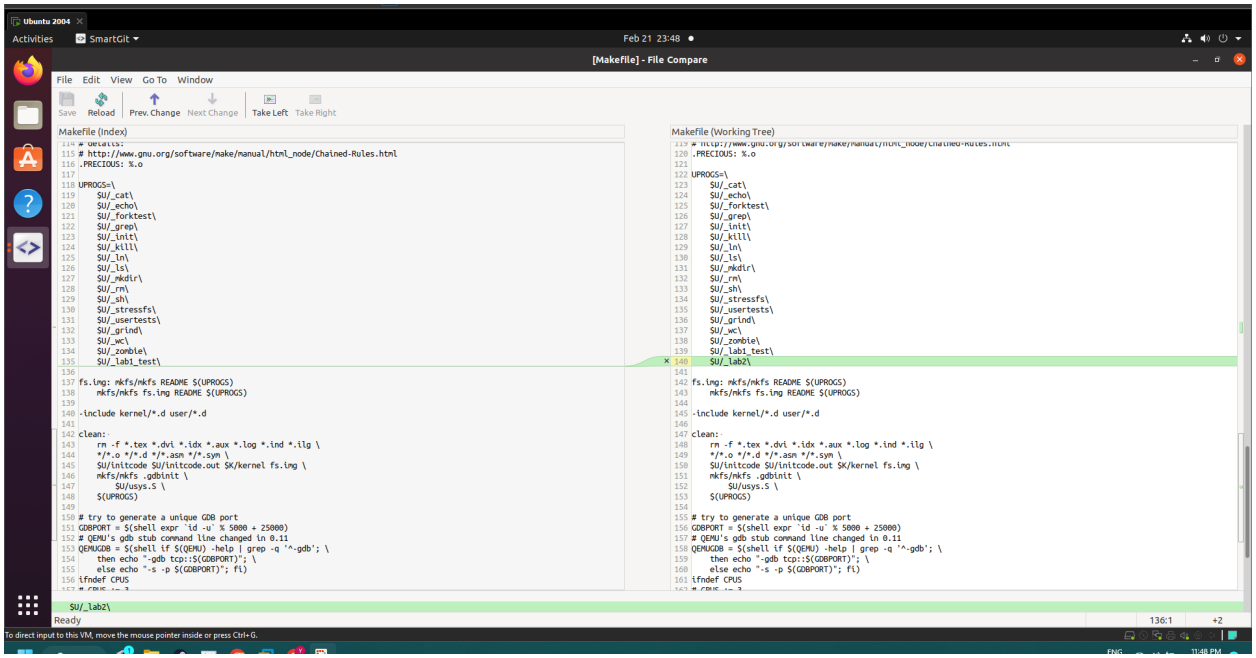

Ubuntu 2004
Activities
SmartGit
Feb 21 23:47
[kernel/proc.c] - File Compare
File Edit View Go To Window
Save Reload Prev. Change Next Change Take Left Take Right
kernel/proc.c (Index)
kernel/proc.c (Working Tree)
#if defined(LOTTERY)

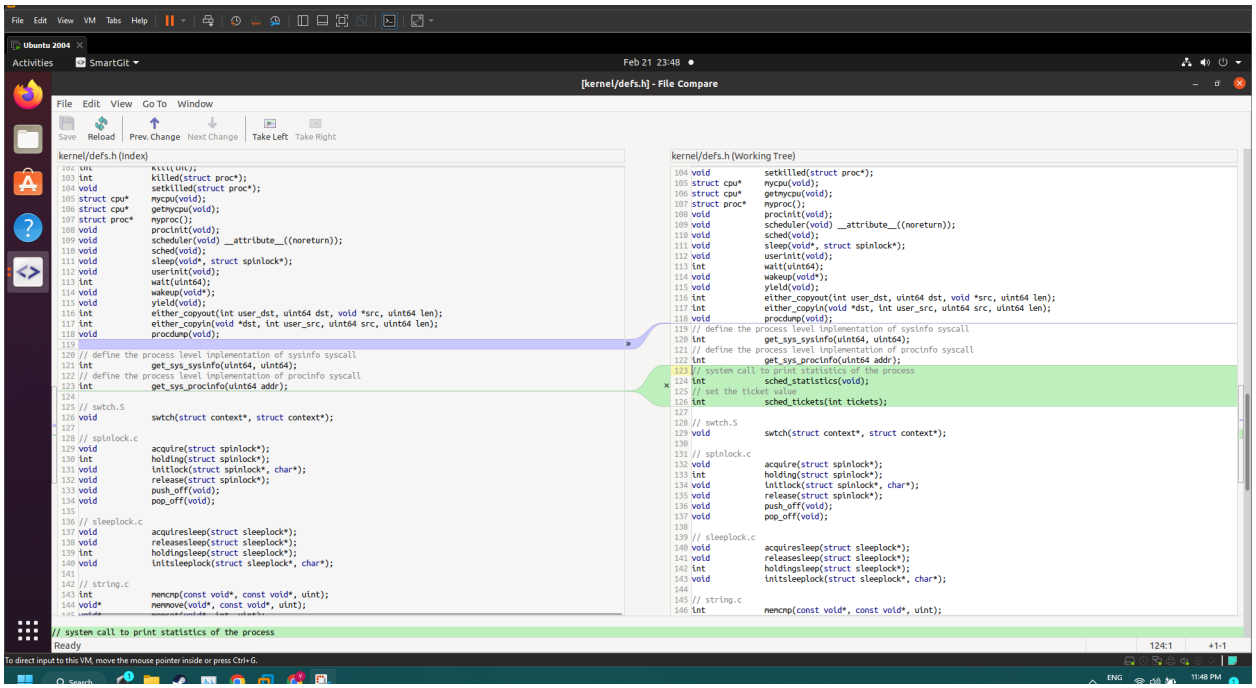

Ubuntu 2004 - VMware Workstation
File Edit View VM Tabs Help
Ubuntu 2004
Activities
SmartGit
Feb 21 23:47
[kernel/proc.c] - File Compare
File Edit View Go To Window
Save
Reload
Prev. Change
Next Change
Take Left
Take Right
kernel/proc.c (Index)
kernel/proc.c (Working Tree)
#if defined(LOTTERY)
Ready
468:1
+3~1
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.

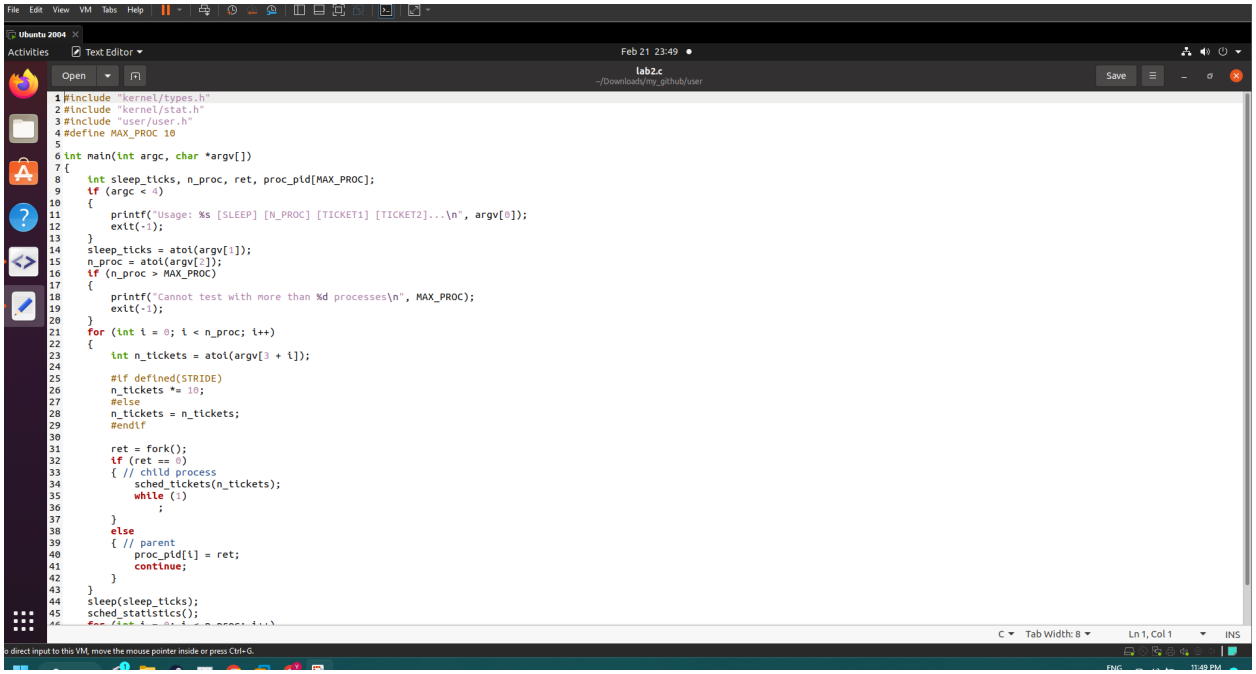

File Edit View VM Tabs Help
Ubuntu 2004
Activities
SmartGit
Feb 21 23:47
[kernel/proc.c] - File Compare
File Edit View Go To Window
Save
Reload
Prev. Change
Next Change
Take Left
Take Right
kernel/proc.c (Index)
kernel/proc.c (Working Tree)
#if defined(LOTTERY)
Ready
468:1
+3~1
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.
ENG
11:47 PM


[kernel/proc.c] - File Compare
kernel/proc.c (Index)
kernel/proc.c (Working Tree)
Ready


[Makefile] - File Compare
Makefile (Index)
Makefile (Working Tree)
Ready


Ubuntu 2004
Activities
SmartGit
Feb 21 23:48
[Makefile] - File Compare
File Edit View Go To Window
Save Reload Prev. Change Next Change Take Left Take Right
Makefile (Index)
Makefile (Working Tree)
$U/_lab2\
Ready
136:1 +2
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.


File Edit View VM Tabs Help
Ubuntu 2004
Activities
SmartGit
Feb 21 23:48
[kernel/defs.h] - File Compare
File Edit View Go To Window
Save Reload Prev. Change Next Change Take Left Take Right
kernel/defs.h (Index)
kernel/defs.h (Working Tree)
// system call to print statistics of the process
Ready
124:1 +1-1
To direct input to this VM, move the mouse pointer inside or press Ctrl+G.




```c
#include "kernel/types.h"
#include "kernel/stat.h"
#include "user/user.h"
#define MAX_PROC 10

int main(int argc, char *argv[])
{
    int sleep_ticks, n_proc, ret, proc_pid[MAX_PROC];
    if (argc < 4)
    {
        printf("Usage: %s [SLEEP] [N_PROC] [TICKET1] [TICKET2]...\n", argv[0]);
        exit(-1);
    }
    sleep_ticks = atoi(argv[1]);
    n_proc = atoi(argv[2]);
    if (n_proc > MAX_PROC)
    {
        printf("Cannot test with more than %d processes\n", MAX_PROC);
        exit(-1);
    }
    for (int i = 0; i < n_proc; i++)
    {
        int n_tickets = atoi(argv[3 + i]);

        #if defined(STRIDE)
        n_tickets *= 10;
        #else
        n_tickets = n_tickets;
        #endif

        ret = fork();
        if (ret == 0)
        { // child process
            sched_tickets(n_tickets);
            while (1)
                ;
        }
        else
        { // parent
            proc_pid[i] = ret;
            continue;
        }
    }
    sleep(sleep_ticks);
    sched_statistics();
    for (int i = 0; i < n_proc; i++)
        kill(proc_pid[i]);
    exit(0);
}
```