trace

- User program with trace systemcall was written in user/strace.c,kernel/defs.h
- Definition of systemcall in user/user.h
- Entry of systemcall is added in user/usys.pl
- Systemcall number **[23]** in kernel/syscall.h

sigint and sigalarm

- User program with sigalarm and sigreturn systemcalls were written in user/sysproc.c,kernel/defs.h
- Definition of systemcalls in user/user.h
- Entry of systemcall is added in user/usys.pl
- Added variables nticks,alarm_work,alarm_trapframe,ticks_curr to struct proc in kernel/proc.h and necessary initializations were done in kernel/proc.c
- Systemcall numbers in kernell/syscall.h
  sigalarm->**[24]**
  sigreturn->**[25]**

FCFS

- selects the process with least creation time.
- Added a variable ctime ,nprocesses to struct proc in kernel/proc.h and initilaized it to ticks in allocproc() function of kernel/proc.c
- Made changes to scheduler() function in kernel/proc.c ,we initialize a new process called process and we are assigning the process to the process with least ctime.
- Then we change the processor to switch from other process to this and increase nprocesses count.
- Disable the preemption of the process after the clock interrupts in kernel/trap.c

PBS

- Selects the process with highest priority of execution.
- When two or more processes have the same priority, the number of times the process has been scheduled is used to determine the priority. In case the tie still remains, the start-time of the processes are used to break the tie, with the processes having a lower start-time being assigned a higher priority.

- Added a variable stat_p ,ticks_running,ticks_sleeping,total_ticks,start,end,ntickets to struct proc in kernel/proc.h and initialized it to 0 in allocproc() function of kernel/proc.c
- Made changes to scheduler() function in kernel/proc.c ,we initialize a new process called process and we are assigning the process to the process with highest priority.
- Made changes to clockintr() function of kernel/proc.c track runtime and wait time.
- Added a new function set_priority and dynpro in kernel/proc.c
- Added sys_set_priority() system call in kernel/sysproc.c and is given system call number **[26]** in kernel/sysproc.h and code for systemcall is written in kernel/sysproc.c

LBS

- Functions set_process_tickets,seed_gen_random, gen_random,random_mod were declared in kernel/proc.h and kernel/defs.h and corresponding codes for functions were written in kernel/proc.c
- Variables ntickets,times_scheduled were declared in proc.h and initialized in kernel/proc.c
- Necessary changes were done in fork in kernel/proc.c for child process to inherit the same number of tickets as its parents.
- Global variable total_tickets is declared in kernel/proc.c (to maintain total tickets of all processes) and necessary operations were done to total_tickets and p->ntickets(p is a process) in sleep() and exit() functions in kernel/proc.c
- Added sys_settickets() system call in kernel/sysproc.c and is given system call number **[27]** in kernel/sysproc.h and code for systemcall is written in kernel/sysproc.c

MLFQ

- Defined macros QCOUNT(5)and QSIZE(NPROC+1) to represent the number of queues and the size of each queue in kernel/param.h
- Created an enum queued, and a variable queue_state in struct proc to store whether the process is in a queue in MLFQ or not.
- Added a new struct called struct queue and defined functions push ,pop,remove(removes the process onto specified queue and change it's queue_state)and empty ,also defined queuetable (struct queue) in kernel/proc.h

- Create a function queuetableinit() to initialize all structs of the queuetable when starting the xv6.
- Also added new variables q_level,q_rtime,q_etime,q_array[QCOUNT] and initialize them to 0 in kernel/proc.c also added get_preempted() function
- Made changes to scheduler() function in kernel/proc.c to schedule processes according to MLFQ policy.
- Made changes to the usertrap and kerneltrap functions in kernel/trap.c also added age_processes to check for priorities.
- Added file setpriority.c to user.This file is to change the pid of given process to given pid

COW

- uvmcopy() function in vm.c is modified in such a way that we will not allocate new pages , we increase the refcnt for the pa using update() function in kernel/kalloc.c
- Kalloc function in kernel/kalloc.c maintains the refcnt for every physical page.
- refcnt will be initialized to 1 since in init_free function,kfree is called and it decreases the refcnt for every pa.
- Update function in kalloc.c increases the refcnt of the pa and kfree function in kernel/kalloc.c decreases the refcnt of the pa.When refcnt of the pa reduces to 0,pa will be freed
- kalloc function will allocate a pa , if the pa refcnt is not valid , panic.

	Average running time	**Average waiting time**
RR	17	111
FCFS	35	42
PBS	18	107
LBS	18	107
MLFQ	15	136