Implementation of SJF:

In the implementation of the SJF, we calculate the process with the minimum estimated CPU burst and schedule it in the scheduler().
The proc structure is equipped with a new field, estimate, which stores the estimated CPU burst of the process at a particular instant when it is being scheduled.
We also maintain 3 global variables, max_burst, mi_burst and sum_burst which store the Maximum burst time, minimum burst time and the sum of all the CPU burts respectively.
The CPU burst, ends when a particular Running process gets context-switched out into the sleep, runnable or zombie state.
The estimations of the next CPU burst are done in the sleep, yield and exit functions.
When the end_ticks and the start_ticks of the process are updated, we update the next estimated CPU burst using the formula:
$s(n+1) = t(n) - (p*t(n))/q + (p*s(n))/q$
This is done for all the processes when they enter either of the states after the ticks are updated and then a loop is iterated over all the processes which are in the batch and are runnable to find the one out with the minimum estimated CPU burst using the global variable min_burst.
Then, in the scheduler() function, the processes are looped over and we search for the process with the next estimated CPU burst equal to the minimum CPU burst obtained, and schedule it.
The statistics corresponding to the SJF implementation are calculated in the exit() fucntion, corresponding to the CPU bursts, estimated CPU bursts and the error in CPU bursts.

Implementation of UNIX:

In the implementation of the UNIX, we calculate the process with the minimum priority value and schedule it in the scheduler().
The proc structure is equipped with two new fields, priority, prio and cpu_usage, which store the current priority of the process, the base priority, and the current cpu usage of the process respectively.
The scheduling is done for the processes which are in the ready queue, and among them, the one with the minimum priority value (maximum priority) is scheduled.
In the implementation, the priorities of the processes are updated in the sleep() and yield() functions.
Only the CPU USAGE of the current process is incremented by SCHED_PARAM_CPU_USAGE or SCHED_PARAM_CPU_USAGE/2, in case it's done in the yield() or sleep() functions respectively.
Also, for all the processes in the batch, which are runnable, the priority values are updated as follows :
CPU usage = (CPU usage)/2
priority = (base priority) + (CPU usage)/2
Also, a global variable min_prio is maintained to update the minimum priority values, whenever the above update takes place.
Then, in the scheduler() function, the processes are looped over and we search for the process with the priority equal to the minimum priority obtained, and schedule it.
The statistics corresponding to the UNIX implementation are calculated in the exit() fucntion.