HW3 | Hamoud abdulaziz alqusair | XXXXXX

Q1)

* New , Running , Ready , Waiting , terminated

Q2)

* because short-term always pick process from ready queue to be executed due to several reason such as when the old process interrupted via time interrupt , I/O interrupt .. etc which occurs frequently while process being executed so CPU need another process to be executed , therefore short-term (CPU scheduler will start picking the next process ) while long-term start only when picking job from job-queue which happen infrequently ( once program is initialized to process )

Q3)

* improve multiprogramming by allowing multiple processes to reside in main memory by swapping out processes that are waiting (need I/O) or low priority processes and swapping in other processes that were in ready queue

Q4)

* init is the root or parent of all process , after system booted init process could create various child’s

Q5)

* because the amount of time wasted ( CPU remains idle ) while context of old process is being saved and the context of the next process being loaded .

Q6)

* a.1) the parent continues to execute concurrently with it’s children
* a.2) the parent waits until some or all of it’s children have terminated
* b.1) the child process is a duplicate of the parent process
* b.2) the child process has new program loaded into it

Q7)

* producer could produce data without checking the buffer whether it’s full or empty and consumer will wait if the buffer is empty till data comes to the buffer .

Q8)

* Information sharing : users may be interested in the same piece of data ( such as shared file ) therefore we must provide such an env to allow concurrent access
* Computation speedup : to run task faster we must break it into subtasks , each will run in parallel ( requires multi-cores )
* Modularity : if we want to implement the system in modular fashion , we must divide functions into separate process and threads
* Convenience : to work in many tasks at the same time

Q9)

* orphan process : If parent process is dead but its child process is alive, the child process is declared orphan ( they will be adopted by init process )
* zombie process : If the child process is dead but its parent process is alive, the child process is declared zombie
* so one is dead because their parent haven’t reap them , while the other are alive but their parent terminated before them .