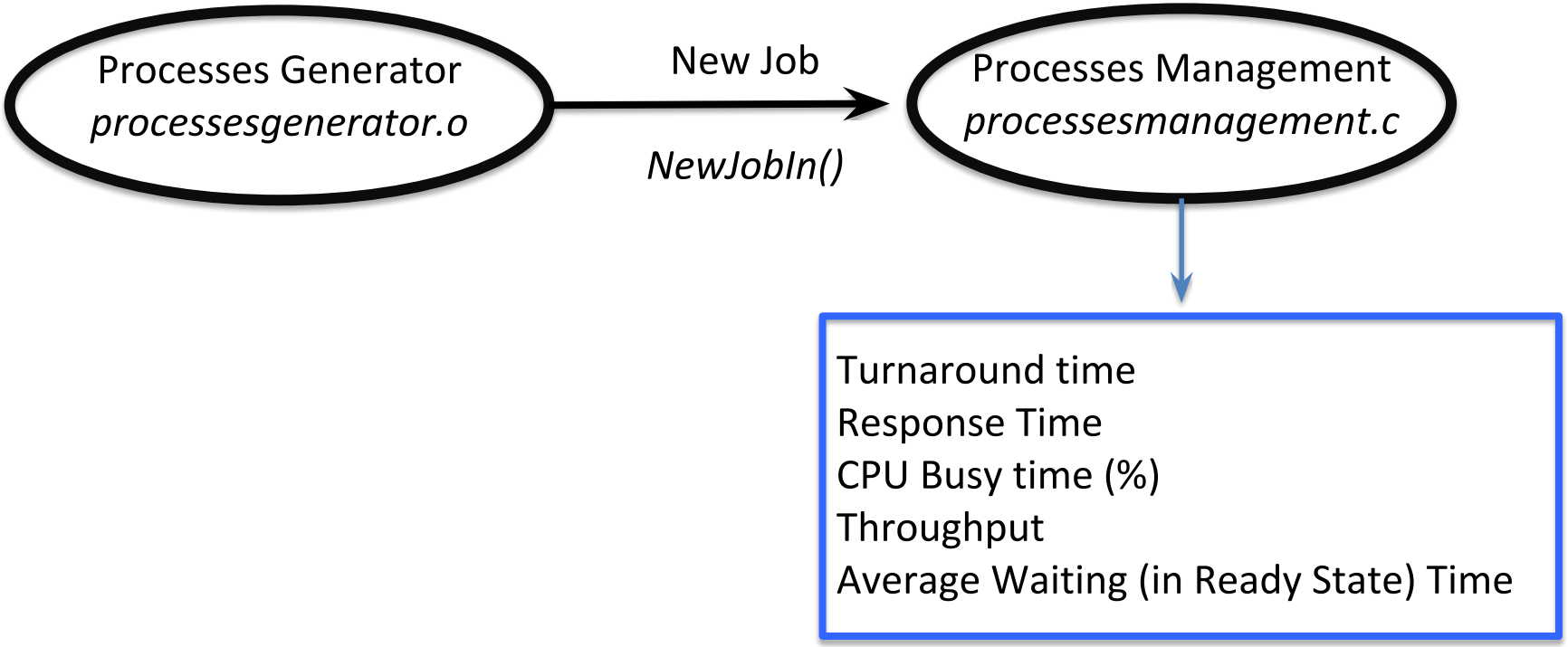
Lab 1 Assignment

Convention:

items in *italic* refer to code

System Overview



Processes Generator

processesgenerator.o

- **Role**: generates processes with inter-arrival exponentially distributed.
- Implemented by instructor *(processesgenerator.o)*
- Whenever a process is generated, the routine ***NewJobIn*** is called.

Processes Management

processesmanagement.c

- **Role**: assigns resources like CPU (lab 1) and memory (lab 2) to **execute** the processes.
- Implemented by students
- **Lab 1**: Evaluate different CPU scheduling policies. CPU will be assigned using FCFS, Shortest Remaining Time First, and RR.
- **Lab 2**: Memory must be assigned using different strategies: TBD.

Process Control Block

common.h

```
typedef struct ProcessControlBlockTag{
  Identifier ProcessID;
  State      state;
  Priority   priority;
  Timestamp  JobArrivalTime;   /* Time when job first entered job queue    */
  TimePeriod TotalJobDuration; /* Total CPU time job requires             */
  TimePeriod TimeInCpu;        /* Total time process spent so far on CPU   */
  TimePeriod CpuBurstTime;     /* Length of typical CPU burst of job       */
  TimePeriod RemainingCpuBurstTime; /* Remaing time of current CPU burst   */
  TimePeriod IOBurstTime;      /* Length of typical I/O burst of job       */
  TimePeriod TimeIOBurstDone;  /* Time when current I/O will be done       */
  Timestamp  JobStartTime;     /* Time when job first entered ready queue  */
  Timestamp  StartCpuTime;     /* Time when job was first placed on CPU    */
  Timestamp  JobExitTime;      /* Time when job first entered exit queue   */
  TimePeriod TimeInReadyQueue; /* Total time process spent in ready queue  */
  TimePeriod TimeInWaitQueue;  /* Total time process spent in wait queue   */
  TimePeriod TimeInJobQueue;   /* Total time process spent in job queue    */
  Memory     TopOfMemory;      /* Address of top of allocated memory block */
  Memory     MemorySize;       /* Amount of allocated memory in bytes      */
  struct ProcessControlBlockTag *previous; /* previous element in linked list */
  struct ProcessControlBlockTag *next;     /* next element in linked list */
} ProcessControlBlock;
```

Queues

typedef enum {JOBQUEUE,READYQUEUE,RUNNINGQUEUE,WAITINGQUEUE,EXITQUEUE} Queue;

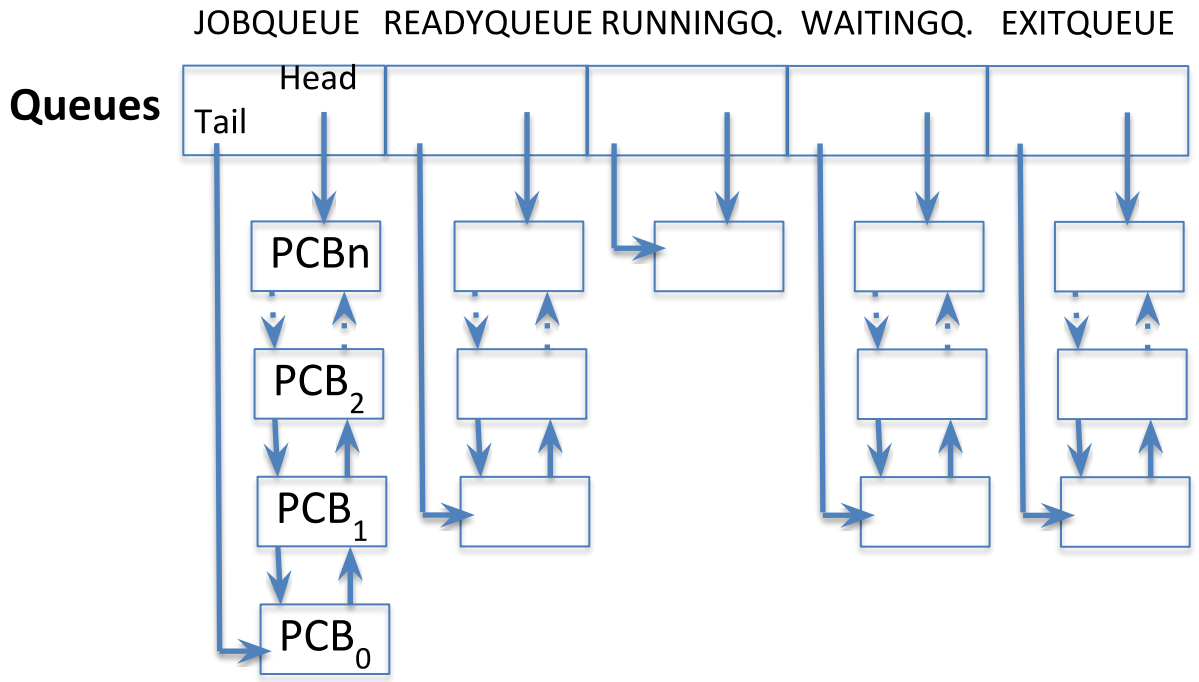
typedef struct QueueParmsTag{

*ProcessControlBlock *Head;*

*ProcessControlBlock *Tail;*

*} **QueueParms**;*

*QueueParms **Queues**[MAXQUEUES];*



Compiling/Executing

- cc -o pm processesgenerator.o processesmanagement.c –lm
- usage: command PolicyNumber [Optional Quantum (in ms)] [Optional Show]

- **PolicyNumber**
  - 1 for First Come First Serve (FCFS)
  - 2 for Shortest Remaining Time First (SRTF)
  - 3 for Round Robin (RR)
- **Quantum**
  - Only for RR
  - Input: 5, 10, 15, 20, 25, 50 …. 500 (in milliseconds)
  - In the program, it is a TimePeriod (float)
- **Show**
  - 1 if you want to see some print messages (e.g., process generation)

Main()

```
int main (int argc, char **argv) {

    if (Initialization(argc,argv)){
      ManageProcesses();
    }
} /* end of main function */
```

NewJobIn(ProcessControlBlock whichProcess)

```
void NewJobIn(ProcessControlBlock whichProcess){
  ProcessControlBlock *NewProcess;

  /* Add Job to the Job Queue */
  NewProcess = (ProcessControlBlock *)
               malloc(sizeof(ProcessControlBlock));
  memcpy(NewProcess,&whichProcess,sizeof(whichProcess));
  EnqueueProcess(JOBQUEUE,NewProcess);
  DisplayQueue("Job Queue in NewJobIn",JOBQUEUE);
  LongtermScheduler(); /* Job Admission     */
}
```

ManageProcesses

```
void ManageProcesses(void){
  while (1) {
  } /* while (1) */
}
```

Common.h

```
/*****************************************************************************\
*                        Function prototypes                                  *
\*****************************************************************************/

/*****************************************************************************\
* Input    : None                                                             *
* Output   : Returns the current system time                                  *
\*****************************************************************************/
extern Timestamp          Now(void);

/*****************************************************************************\
 * Input : Queue where to enqueue and Element to enqueue                      *
 * Output: Updates Head and Tail as needed                                    *
 * Function: Enqueues FIFO element in queue and updates tail and head         *
\*****************************************************************************/
extern void               EnqueueProcess(Queue whichQueue,
                              ProcessControlBlock *whichProcess);

/*****************************************************************************\
 * Input : Queue from which to dequeue                                        *
 * Output: Tail of queue                                                      *
 * Function: Removes tail element and updates tail and head accordingly *
\*****************************************************************************/
extern ProcessControlBlock *DequeueProcess(Queue whichQueue);
```

Common.h (Cont'd)

```
/*****************************************************************************\
 * Input : none                                                              *
 * Output: None                                                              *
 * Function: CPU executes whichProcess for CPUBurst                          *
\*****************************************************************************/
extern void             OnCPU(ProcessControlBlock *whichProcess, TimePeriod CPUBurst);
```

Accessing tux Machines

- Use any ssh client to access

  `gate.eng.auburn.edu`

- Unix systems/ Mac terminal
  - Type `ssh username@gate.eng.auburn.edu`

- Windows: use SecureCRT

First Steps on Tux machines

- After you log in …
  1. mkdir 3500 // create a directory named 3500
  2. cd 3500 // get in directory 3500
  3. mkdir labs
  4. cd labs
  5. mkdir lab1
  6. cd lab 1 // get in directory lab1

  Now you are in your working directory… start editing your programs …

Moving Files (local to Tux Machines)

- Tux machines use your H drive
- Use any sftp client to access sftp.eng.auburn.edu