BACS1024 INTRODUCTION TO COMPUTER SYSTEMS

Chapter 10: Processor Management

0. Overview

1. Processes
2. Process State & Transition
3. Process Scheduler
4. Context Switch
5. Process Scheduling Policies
6. Process Scheduling Algorithms (FCFS, SJF, SRT, RR, PS)

1. Processes

1. Processes

- **Processes**
  - Each executing task is known as a process. A process includes a program with all associated resources, such as I/O devices, memory, CPU time, etc
  - In a multitasking environment, OS creates a data block, named Process Control Block (**PCB**) to keep track of each of the different processes that are executing concurrently in memory.
    - PCB contains all relevant information about the process.
    - PCB is used by various OS modules as they perform process-related functions.
    - Contents of each job's PCB: (1) Process identification, (2) Process status (HOLD, READY, RUNNING, WAITING) , (3) Process state (process status word, register contents, main memory info, resources, process priority) & (4) Accounting (CPU time, total amount of time, I/O operations, number input records read, etc.)

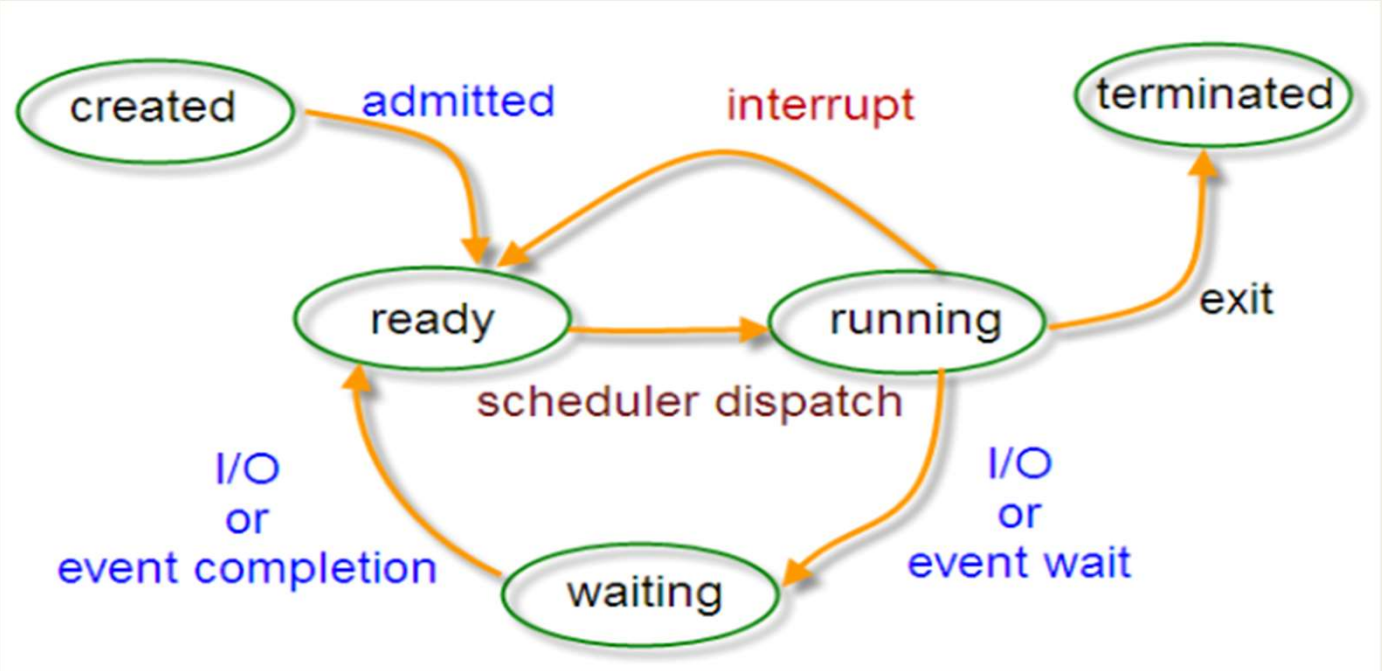
1. Processes

- **Processes**
  - **PCBs and Queuing**
    - **PCB** of job created when Job Scheduler accepts it
      - ✔ It updated as job goes from beginning to termination.
    - **Queues** use PCBs to track jobs.
      - ✔ PCBs, not jobs, are linked to form queues.
      - ✔ E.g., PCBs for every ready job are linked on READY queue; all PCBs for jobs just entering system are linked on HOLD queue.
      - ✔ Queues must be managed by process scheduling policies and algorithms.

2. Process State & Transition

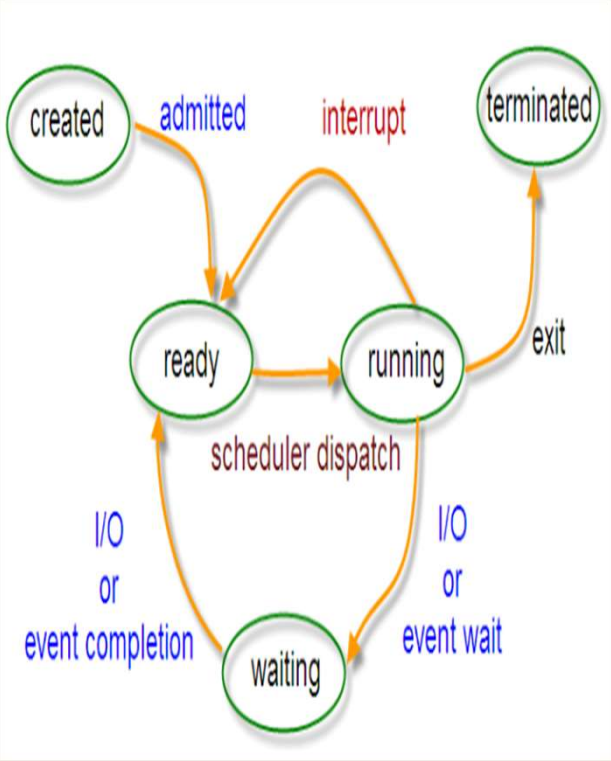
2. Process State & Transition

- **Process / job State**
  - A process / job moves through the system via the following states.
    1. Created
    2. Ready
    3. Waiting
    4. Running
    5. Terminated



2. Process State & Transition

- **Process / job State**



State	Operation
Created to Ready	▪ The process has been created, became a job & admitted ready queue in the system for execution
Ready to Running	▪ The process compete with all other processes in ready state for CPU execution time ▪ The process in the running state has control of the CPU & able to execute instruction
Running to Ready	▪ The OS returns the process to ready queue & await further time for processing
Running to Waiting	▪ The process remain waiting for I/O or other event
Waiting to Ready	▪ Once I/O is completed, the OS moves the process to ready
Running to Terminated	▪ When the process completes execution, controls returns to the OS & the process is terminated

3. Process Schedulers

3. Process Schedulers

- **In the multitasking environment,**
  - Multiple processes are allowed to be executed concurrently
  - The CPU scheduling provides mechanisms for the acceptance of processes into the system & for the actual allocation of CPU time for execution
    - Processor must be allocated to each job in a fair and efficient manner.
    - Requires scheduling policy and a scheduling algorithm.
  - To do these **schedulers** are involved
    - To perform job scheduling, process scheduling & interrupt management.

3. Process Schedulers

- **Schedulers**
  1. High level scheduler
  2. Middle level scheduler
  3. Low level scheduler

3. Process Schedulers

- **Schedulers**

1. **High level scheduler**
   - ❑ Also known as **job scheduler**
   - ❑ Selects jobs from a queue of incoming jobs.
     - ❖ Places them in process queue (batch or interactive), based on each job's characteristics.
     - ❖ Goal is to put jobs in a sequence that uses all system's resources as fully as possible.
     - ❖ Strives for balanced mix of jobs with large I/O interaction and jobs with lots of computation.
   - ❑ Tries to keep most system components busy most of time.

3. Process Schedulers

- **Schedulers**

  2. **Middle level scheduler**

     - In a highly interactive environment there's a third layer, called **middle-level scheduler**.
     - It removes active jobs from memory to reduce degree of multiprogramming and allows jobs to be completed faster.
     - However, problem occurred. **Thrashing** happens when a process is busy swapping pages in and out.

3. Process Schedulers

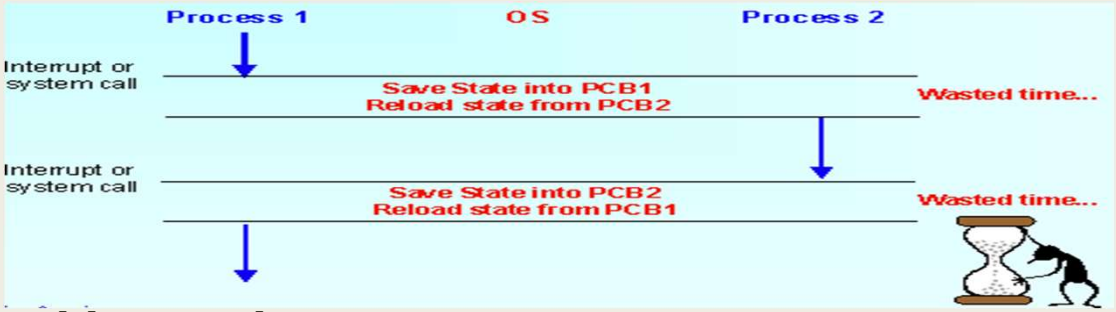
- **Schedulers**

3. Low level scheduler

- Also known as **job scheduler**
- It assigns the CPU to execute processes of those jobs placed on ready queue by Job Scheduler.
- After a job has been placed on the **READY queue** by Job Scheduler, Process Scheduler that takes over.
  - Determines which jobs will get CPU, when, and for how long.
  - Decides when processing should be interrupted.
  - Determines queues job should be moved to during execution.
  - Recognizes when a job has concluded and should be terminated.

4. Context Switch

4. Context Switch



- **In the multitasking environment,**
  - To facilitate effective CPU manipulation, processes are performed at almost concurrently, i.e. the processes share CPU times.
  - In the process of switching the CPU to another (context switch), it requires saving the state of the old process and loading the saved state for the new process.
- **Context switching** is the acts of saving a job's processing information in its PCB so the current job can be swapped out of memory and loads the processing information from the PCB of another new coming or interrupted job into the appropriate registers.

4. Context Switch

- **Operation of Context Switch**

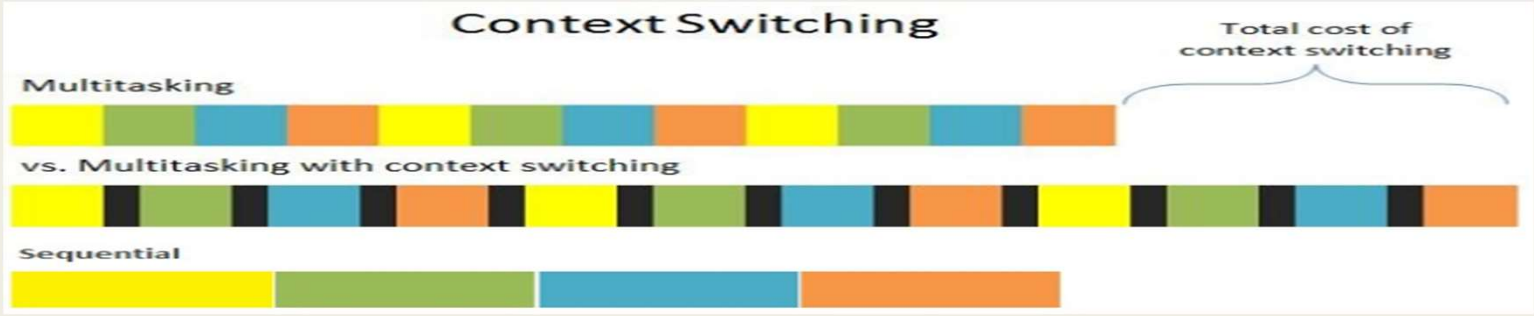
  (a) When a job is preempted

  - ❑ All of its processing information must be saved in its PCB for later (when Job A's execution is continued).
  - ❑ Contents of Job B's PCB are loaded into appropriate registers so it can start running again. (Context switch occurred).

  (b) When Job A is once again assigned to processor

  - ❑ Info from preempted job is stored in its PCB.
  - ❑ Contents of Job A's PCB are loaded into appropriate registers. (Context switch occurred)

4. Context Switch



- **Pros**
  - ✔ Hardware mechanism saves almost all of the CPU state
  - ✔ Software can be more selective and save only that portion that actually needs to be saved & reloaded.
  - ✔ It permits better control over the validity of the data that is being loaded
- **Cons**
  - ✔ It requires considerable processor time

4. Context Switch

- **Solutions**
  - ❑ Linux & some other Unix-like systems, is its extremely low cost of context switching.
  - ❑ The increasing clock speed of CPU has declined the costs of context switching.

- **Usage**
  - ❑ To support ALL Preemptive Algorithms

6. Process Scheduling Policies

5. Process Scheduling Policies

- Before operating system can schedule all jobs in a multiprogramming environment, it must resolve three **limitations of system**:
  - ❑ Finite number of resources (such as disk drives, printers, and tape drives)
  - ❑ Some resources can't be shared once they're allocated (such as printers)
  - ❑ Some resources require operator intervention (such as tape drives).

5. Process Scheduling Policies

Guidelines to GOOD Process Scheduling Policies

1) Maximize throughput by running as many jobs as possible in a given amount of time.
2) Maximize CPU efficiency by keeping CPU busy 100 % of time.
3) Ensure fairness for all jobs by giving everyone an equal amount of CPU and I/O time.
4) Minimize response time by quickly turning around interactive requests.
5) Minimize turnaround time by moving entire jobs in/out of system quickly.
6) Minimize waiting time by moving jobs out of READY queue as quickly as possible.

5. Process Scheduling Policies

- **Types of Process Scheduling Policies:**

1. **Preemptive scheduling policy**
   - ❑ Interrupts processing of a job and transfers the CPU to another job.
   - ❑ Includes various algorithms: (a) Shortest Remaining Time (SRT)
     (b) Round Robin (RR)
     (c) Priority Scheduling (PS)

2. **Non-preemptive scheduling policy**
   - ❑ Process scheduling functions without external interrupts.
   - ❑ Includes various algorithms: (a) First Come First Serve (FCFS)
     (b) Shortest Job First (SJF)
     (c) Priority Scheduling (PS) - default

6. Process Scheduling Algorithms

6. Process Scheduling Algorithms

(a) **First Come First Served (FCFS)**

- Non-preemptive.
- Handles jobs according to their arrival time. The earlier they arrive, the sooner they're served.
- Simple algorithm to implement. Uses a FIFO queue.
- Good for batch systems; not so good for interactive ones.
- **Turnaround time** is unpredictable.

6. Process Scheduling Algorithms

(b) **Shortest Job Next (SJN)**

- Non-preemptive.
- Handles jobs based on length of their CPU cycle time. Use lengths to schedule process with shortest time.
- Optimal – gives minimum average waiting time for a given set of processes. Optimal only when all of jobs are available at same time and the CPU estimates are available and accurate.
- Doesn't work in interactive systems because users don't estimate in advance CPU time required to run their jobs.

6. Process Scheduling Algorithms

(c) **Shortest Remaining Time (SRT)**

- Preemptive version of the SJN algorithm.
- Processor allocated to job closest to completion. This job can be preempted if a newer job in READY queue has a “time to completion” that's shorter.
- Can’t be implemented in interactive system -- requires advance knowledge of CPU time required to finish each job.
- SRT involves more overhead than SJN. OS monitors CPU time for all jobs in READY queue and performs “context switching”.

6. Process Scheduling Algorithms

(d) **Round Robin**

- Used extensively in interactive systems because it's easy to implement.
- Isn't based on job characteristics but on a predetermined slice of time that's given to each job. Ensures CPU is equally shared among all active processes and isn't monopolized by any one job.
- **Time slice** is called a **time quantum** . Size crucial to system performance
- **If Job's CPU Cycle < Time Quantum**
  - If job's last CPU cycle & job is finished, then all resources allocated to it are released & completed job is returned to user.
  - If CPU cycle was interrupted by I/O request, then info about the job is saved in its PCB & it is linked at end of the appropriate I/O queue. Later, when I/O request has been satisfied, it is returned to end of READY queue to await allocation of CPU.
- **Time Slices Should Be ...**
  - Long enough to allow 80 % of CPU cycles to run to completion.

6. Process Scheduling Algorithms

(e) **Priority Scheduling**

- Non-preemptive algorithm which is commonly used in batch systems.
- Gives preferential treatment to important jobs.
- Allows the program with the highest priority to be processed first and these high priority jobs are not interrupted until their CPU cycles (run times) are completed or a *natural wait* occurs.
- If two or more jobs with equal priority, then uses FCFS policy within the same priority group
- Note:
  - *Natural wait* is a common term used to identify an I/O request from a program in a multiprogramming environment that would cause a process to wait “naturally” before resuming execution.

6. Process Scheduling Algorithms

Policy	Algorithm	Based on	Pros	Cons
Non-preemptive	First Come First Serve (FCFS)	Arrival time (FIFO)	Batch system	Interactive system
Non-preemptive	Shortest Job First (SJN)	Based on shortest CPU cycle	Job available at same time	Interactive system
Non-preemptive / Preemptive	Priority scheduling (PS)	Based on priority & FCFS	Preferential system	low priority job keeps waiting
Preemptive	Shortest remaining time (SRT)	Based on shortest remaining CPU cycle	Fastest completion	Interactive system
Preemptive	Round robin (RR)	Based on quantum	Equally share CPU	More overhead
Wait Time = Finish Time – CPU Cycle – Arrival Time Turnaround Time = Finish Time – Arrival Time				

6. Process Scheduling Algorithms

- **To draw timeline and to do calculations for process scheduling algorithms**
  - E.g.:

    Given the following information, draw a timeline for each of the following algorithms. Also calculate the average waiting time and average turnaround time for each of them.

    (a) FCFS (First Come First Served)

    (b) SJN (Shortest Job Next)

    (c) SRT (Shortest Remaining Time)

    (d) Round Robin (Time Quantum = 3)

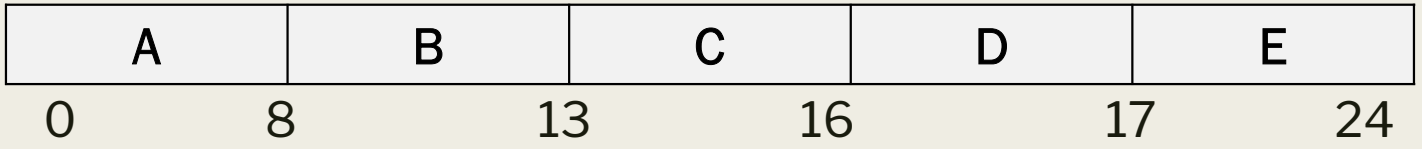
    (e) Priority scheduling

  - **Wait Time = Finish Time – CPU Cycle – Arrival Time**
  - **Turnaround Time = Finish Time – Arrival Time**

Job	AT	CT	P
A	0	8	5
B	1	5	3
C	2	3	1
D	3	1	4
E	4	7	2

6. Process Scheduling Algorithms

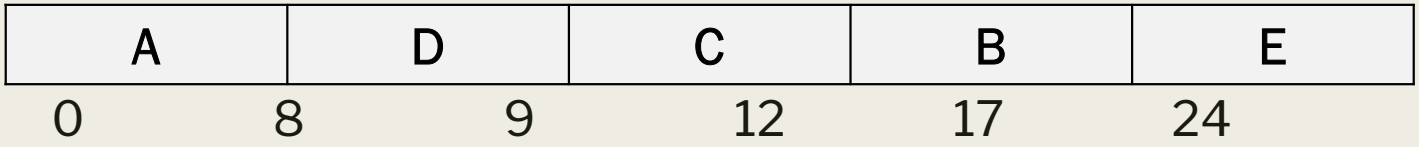
(a) FCFS (First Come First Served)



Job	AT	CT	FT	TT	WT
A	0	8	8	8	0
B	1	5	13	12	7
C	2	3	16	14	11
D	3	1	17	14	13
E	4	7	24	20	13
	Total:	24	Total:	68	44
			Average:	13.6	8.8

6. Process Scheduling Algorithms

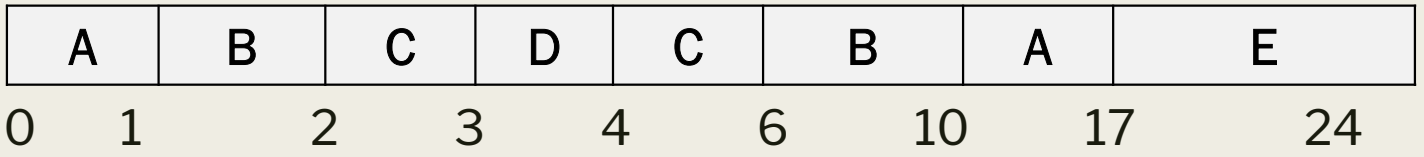
(b) SJN



Job	AT	CT	FT	TT	WT
A	0	8	8	8	0
B	1	5	17	16	11
C	2	3	12	10	7
D	3	1	9	6	5
E	4	7	24	20	13
	Total:	24	Total:	60	36
			Average:	12	7.2

6. Process Scheduling Algorithms

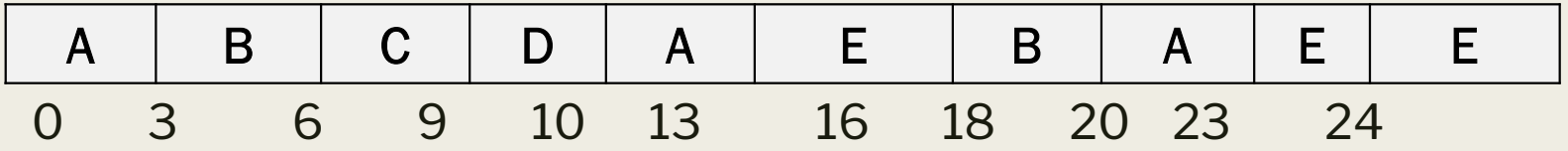
(c) SRT



Job	AT	CT	FT	TT	WT
A	0	8	17	17	9
B	1	5	10	9	4
C	2	3	6	4	1
D	3	1	4	1	0
E	4	7	24	20	13
	Total:	24	Total:	51	27
			Average:	10.2	5.4

6. Process Scheduling Algorithms

(d) RR (Q=3)



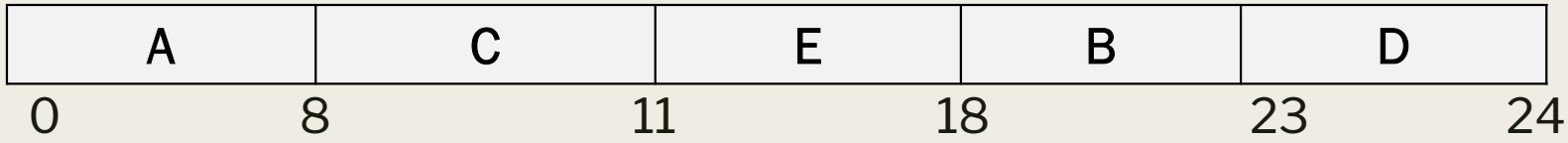
Job	AT	CT	FT	TT	WT
A	0	8	20	20	12
B	1	5	18	17	12
C	2	3	9	7	4
D	3	1	10	7	6
E	4	7	24	20	13
	Total:	24	Total:	71	47
			Average:	14.2	9.4

Ready Queue

~~B,C,D,A,E,B,A,~~
~~E,E~~

6. Process Scheduling Algorithms

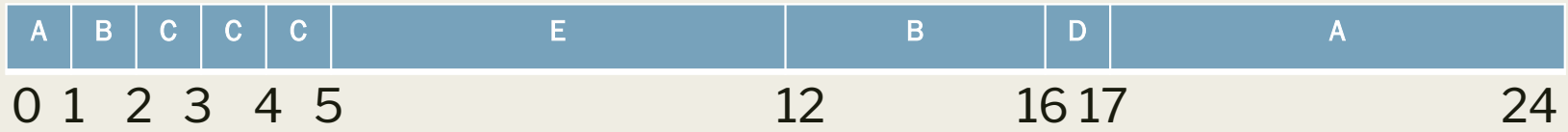
(e) PS (Non-Preemptive)



Job	AT	CT	P	FT	TT	WT
A	0	8	5	8	8	0
B	1	5	3	23	22	17
C	2	3	1	11	9	6
D	3	1	4	24	21	20
E	4	7	2	18	14	7
	Total:	24		Total:	74	50
				Average:	14.8	10

6. Process Scheduling Algorithms

(e) PS (Preemptive)



Job	AT	CT	P	FT	TT	WT
A	0	8	5	24	24	16
B	1	5	3	16	15	10
C	2	3	1	5	3	0
D	3	1	4	17	14	13
E	4	7	2	12	8	1
	Total:	24		Total:	64	40
				Average:	12.8	8.0

Chapter Review

Chapter Review

1. **Processes**
2. **Process State & Transition**
   - ❑ Create
   - ❑ Ready
   - ❑ Running
   - ❑ Waiting
   - ❑ Terminate
3. **Scheduler**
   - ❑ High level
   - ❑ Middle level
   - ❑ Low level
4. **Context Switch**
   - ❑ Pros & cons
   - ❑ Solutions
5. **Process Scheduling Policies**
   - ❑ Preemptive scheduling policies
   - ❑ Non-preemptive scheduling policies
6. **Process Scheduling Algorithms**
   - ❑ FCFS
   - ❑ SJF
   - ❑ SRT
   - ❑ RR
   - ❑ PS