21BDS0340

Abhinav Dinesh Srivatsa

Operating Systems Lab

Assignement – III

Practice:

- Finding area and circumference of a circle

Program:

```
echo Enter radius:
read n
area=`expr $n \* $n \* 22 / 7`
circ=`expr 2 \* $n \* 22 / 7`
echo Area is $area
echo Circumference is $circ
```

Output:

```
(base) abhi@Abhinavs-MacBook-Pro Assignment 3 % bash practice1.sh
Enter radius:
3
Area is 28
Circumference is 18
```

- Finding the sum of the first 5 natural numbers

Program:

```
sum=0
echo First 5 natural numbers
for i in 1 2 3 4 5
do
sum=`expr $sum + $i`
echo $i
done
echo The sum is $sum
```

Output:

```
(base) abhi@Abhinavs-MacBook-Pro Assignment 3 % bash practice2.sh
First 5 natural numbers
1
2
3
4
5
The sum is 15
```

Assignment:

Question 1

- First come first served (FCFS)

Program:

```
arrival=()
burst=()

echo Enter arrival times:
read input
read -a arrival <<< "$input"

echo Enter burst times:
read input
read -a burst <<< "$input"

if [ ${#arrival[@]} -eq ${#burst[@]} ]
then
    arrival_len=${#arrival[@]}

    # selection sort by arrival time
    for((i=0; i<arrival_len-1; i++));
    do
        min=$i
        for ((j=i+1; j<arrival_len; j++))
        do
            if ((arrival[j] < arrival[min]))
            then min=$j
            fi
        done

        # swapping arrival time
        temp=${arrival[i]}
        arrival[i]=${arrival[min]}
        arrival[min]=$temp

        # swapping burst times
        temp=${burst[i]}
        burst[i]=${burst[min]}
        burst[min]=$temp
    done

    time=${arrival[0]}

    # display no process if time doesnt start at 0
    if ((time != 0))
    then echo 0 - $time: No process in ready queue
    fi
```

```
    # display results
    for i in ${!arrival[@]}
    do
        echo $time - $((time + burst[i])): Process with arrival time: ${arrival[i]}
and burst time: ${burst[i]}
        time=$((time + burst[i]))
    done

else echo Number of values must be equal!
fi
```

Output:

```
(base) abhi@Abhinavs-MacBook-Pro Assignment 3 % bash question1fcfs.sh
Enter arrival times:
5 4 3 0 3
Enter burst times:
5 2 7 4 5
0 - 4: Process with arrival time: 0 and burst time: 4
4 - 11: Process with arrival time: 3 and burst time: 7
11 - 16: Process with arrival time: 3 and burst time: 5
16 - 18: Process with arrival time: 4 and burst time: 2
18 - 23: Process with arrival time: 5 and burst time: 5
```

- Shortest job first (SJF)

Program:

```
arrival=()
burst=()
ready=()
completed=()

echo Enter arrival times:
read input
read -a arrival <<< "$input"

echo Enter burst times:
read input
read -a burst <<< "$input"

if [ ${#arrival[@]} -eq ${#burst[@]} ]
then
    arrival_len=${#arrival[@]}

    # selection sort by arrival time then burst time
    for((i=0; i<arrival_len-1; i++));
    do
        min=$i
        for ((j=i+1; j<arrival_len; j++))
        do
            if ((arrival[j] < arrival[min]))
            then min=$j
            fi
```

```
        if ((arrival[j] == arrival[min]))
        then if ((burst[j] < burst[min]))
            then min=$j
            fi
        fi
    done

    # swapping arrival time
    temp=${arrival[i]}
    arrival[i]=${arrival[min]}
    arrival[min]=$temp

    # swapping burst times
    temp=${burst[i]}
    burst[i]=${burst[min]}
    burst[min]=$temp
done

time=${arrival[0]}

# looping times for number of processes
for ((n=0; n<arrival_len; n++))
do
    # building ready queue
    for ((i=0; i<arrival_len; i++))
    do
        if ((arrival[i] <= time))
        then
            # checking if process is marked completed or already
            process_complete=0
            completed_len=${#completed[@]}
            for ((j=0; j<completed_len; j++))
            do
                if ((i == completed[j]))
                then process_complete=1
                fi
            done
            if ((process_complete == 0))
            then ready+=($i)
            fi
        fi
    done

    # finding min burst time in ready queue
    min_burst=0
    ready_len=${#ready[@]}

    for ((i=0; i<ready_len; i++))
    do
        if ((burst[ready[i]] < burst[ready[min_burst]]))
        then min_burst=$i
```

```
            fi
        done

        # displaying process info
        echo $time - $((time + burst[ready[min_burst]])): Process with arrival
time: ${arrival[ready[min_burst]]} and burst time: ${burst[ready[min_burst]]}
        time=$((time + burst[ready[min_burst]]))

        # marking process as completed
        completed+=($((ready[min_burst])))

        # clearing ready queue
        ready=()
    done

else echo Number of values must be equal!
fi
```

Output:

```
(base) abhi@Abhinavs-MacBook-Pro Assignment 3 % bash question1sjf.sh
Enter arrival times:
5 4 3 0 3
Enter burst times:
5 2 7 4 5
0 - 4: Process with arrival time: 0 and burst time: 4
4 - 6: Process with arrival time: 4 and burst time: 2
6 - 11: Process with arrival time: 3 and burst time: 5
11 - 16: Process with arrival time: 5 and burst time: 5
16 - 23: Process with arrival time: 3 and burst time: 7
```

- Priority

Program:

```
arrival=()
burst=()
priority=()
ready=()
completed=()

echo Enter arrival times:
read input
read -a arrival <<< "$input"

echo Enter burst times:
read input
read -a burst <<< "$input"

echo Enter priority times:
read input
read -a priority <<< "$input"

if [ ${#arrival[@]} -eq ${#burst[@]} ]
```

```
then
    arrival_len=${#arrival[@]}

    # selection sort by priority and arrival time
    for((i=0; i<arrival_len-1; i++));
    do
        min=$i
        for ((j=i+1; j<arrival_len; j++))
        do
            if ((priority[j] < priority[min]))
            then min=$j
            fi
            if ((priority[j] == priority[min]))
            then if ((arrival[j] < arrival[min]))
                then min=$j
                fi
            fi
        done

        # swapping arrival time
        temp=${arrival[i]}
        arrival[i]=${arrival[min]}
        arrival[min]=$temp

        # swapping burst times
        temp=${burst[i]}
        burst[i]=${burst[min]}
        burst[min]=$temp

        # swapping priorities
        temp=${priority[i]}
        priority[i]=${priority[min]}
        priority[min]=$temp
    done

    time=${arrival[0]}

    # looping times for number of processes
    for ((n=0; n<arrival_len; n++))
    do
        # building ready queue
        for ((i=0; i<arrival_len; i++))
        do
            if ((arrival[i] <= time))
            then
                # checking if process is marked completed or already
                process_complete=0
                completed_len=${#completed[@]}
                for ((j=0; j<completed_len; j++))
                do
                    if ((i == completed[j]))
```

```
                    then process_complete=1
                    fi
                done
                if ((process_complete == 0))
                then ready+=($i)
                fi
            fi
        done

        # finding min burst time in ready queue
        min_burst=0
        ready_len=${#ready[@]}

        for ((i=0; i<ready_len; i++))
        do
            if ((priority[ready[i]] > priority[ready[min_burst]]))
            then min_burst=$i
            fi
        done

        # displaying process info
        echo $time - $((time + burst[ready[min_burst]])): Process with arrival
time: ${arrival[ready[min_burst]]} and burst time: ${burst[ready[min_burst]]} and
priority: ${priority[ready[min_burst]]}
        time=$((time + burst[ready[min_burst]]))

        # marking process as completed
        completed+=($((ready[min_burst])))

        # clearing ready queue
        ready=()
    done

else echo Number of values must be equal!
fi
```

Output:

```
(base) abhi@Abhinavs-MacBook-Pro Assignment 3 % bash question1prior.sh
Enter arrival times:
5 4 3 0 3
Enter burst times:
5 2 7 4 5
Enter priority times:
0 7 5 10 5
5 - 9: Process with arrival time: 0 and burst time: 4 and priority: 10
9 - 11: Process with arrival time: 4 and burst time: 2 and priority: 7
11 - 18: Process with arrival time: 3 and burst time: 7 and priority: 5
18 - 23: Process with arrival time: 3 and burst time: 5 and priority: 5
23 - 28: Process with arrival time: 5 and burst time: 5 and priority: 0
```

- Round Robin (non-preemptive)

Program:

```
arrival=()
burst=()

echo Enter arrival times:
read input
read -a arrival <<< "$input"

echo Enter burst times:
read input
read -a burst <<< "$input"

if [ ${#arrival[@]} -eq ${#burst[@]} ]
then
    arrival_len=${#arrival[@]}

    # selection sort by arrival time
    for((i=0; i<arrival_len-1; i++));
    do
        min=$i
        for ((j=i+1; j<arrival_len; j++))
        do
            if ((arrival[j] < arrival[min]))
            then min=$j
            fi
        done

        # swapping arrival time
        temp=${arrival[i]}
        arrival[i]=${arrival[min]}
        arrival[min]=$temp

        # swapping burst times
        temp=${burst[i]}
        burst[i]=${burst[min]}
        burst[min]=$temp
    done

    time=${arrival[0]}

    # display no process if time doesnt start at 0
    if ((time != 0))
    then echo 0 - $time: No process in ready queue
    fi

    # display results
    for i in ${!arrival[@]}
    do
```

```
        echo $time - $((time + burst[i])): Process with arrival time: ${arrival[i]}
and burst time: ${burst[i]}
        time=$((time + burst[i]))
    done

else echo Number of values must be equal!
fi
```

Output:

```
(base) abhi@Abhinavs-MacBook-Pro Assignment 3 % bash question1fcfs.sh
Enter arrival times:
5 4 3 0 3
Enter burst times:
5 2 7 4 5
0 - 4: Process with arrival time: 0 and burst time: 4
4 - 11: Process with arrival time: 3 and burst time: 7
11 - 16: Process with arrival time: 3 and burst time: 5
16 - 18: Process with arrival time: 4 and burst time: 2
18 - 23: Process with arrival time: 5 and burst time: 5
```

Question 2

Program

```
echo Enter number of processes:
read processes
echo Enter types of resources:
read resource_types

free_resources=()
assigned=()
required=()

# reading free resources
echo Enter free resources:
read input
read -a free_resources <<< $input

# reading assigned resources
for ((i=0; i<processes; i++))
do
    echo "Enter space-separated assigned resources for process $((i + 1)):"
    read -ra row_input
    assigned+=(${row_input[@]})
done

# reading required resources
for ((i=0; i<processes; i++))
do
    echo "Enter space-separated required resources for process $((i + 1)):"
    read -ra row_input
    required+=(${row_input[@]})
done

completed=()
for ((n=0; n<processes; n++))
do
    # checking if enough resources are available for process
    for ((i=0; i<processes; i++))
    do
        # checking if process already completed
        is_completed=0
        for j in ${completed[@]}
        do
            if ((j == i))
            then is_completed=1
            fi
        done


        if ((is_completed == 0))
        then
```

```
            completable=1
            for ((x=0; x<resource_types; x++))
            do
                r=`expr $resource_types \* $i + $x`
                if ((assigned[r] + free_resources[x] < required[r]))
                then completable=0
                fi
            done
            if ((completable == 1))
            then
                for ((x=0; x<resource_types; x++))
                do
                    free_resources[x]=$((free_resources[x] + assigned[r]))
                done
                completed+=($i)
            fi
        fi
    done
done

completed_len=${#completed[@]}

if ((completed_len == processes))
then echo This state is safe and the sequence is ${completed[@]}
else echo This state is unsafe
fi
```

Output:

```
(base) abhi@Abhinavs-MacBook-Pro Assignment 3 % bash question2.sh
Enter number of processes:
5
Enter types of resources:
3
Enter free resources:
3 3 2
Enter space-separated assigned resources for process 1:
0 1 0
Enter space-separated assigned resources for process 2:
2 0 0
Enter space-separated assigned resources for process 3:
3 0 2
Enter space-separated assigned resources for process 4:
2 1 1
Enter space-separated assigned resources for process 5:
0 0 2
Enter space-separated required resources for process 1:
7 5 3
Enter space-separated required resources for process 2:
3 2 2
Enter space-separated required resources for process 3:
9 0 2
Enter space-separated required resources for process 4:
2 2 2
Enter space-separated required resources for process 5:
4 3 3
This state is safe and the sequence is 1 3 4 2 0
```

Question 3

Program:

```
semaphore=1

wait(){
    while ((semaphore <= 0))
    do semaphore=$semaphore # do nothing
    done
    semaphore=$((semaphore - 1))
    sleep 5 # simulating process working
    signal
}

signal(){
    semaphore=$((semaphore + 1))
}

# executing process 1
echo Started process 1...
wait
echo Process 1 completed.
echo ""

# wait and then execute process 2
echo Started process 2...
wait
echo Process 2 completed.
```

Output:

```
(base) abhi@Abhinavs-MacBook-Pro Assignment 3 % bash question3.sh
Started process 1...
Process 1 completed.

Started process 2...
Process 2 completed.
(base) abhi@Abhinavs-MacBook-Pro Assignment 3 %
```