Operating System Lab Manuals

Subject: Operating Systems Lab Code: AMC15206 L T P: 0-0-3

1. Introduction to Shell Programming
2. Syntax, various commands, algorithm for Shell Programming
3. Execution of Shell Programming
4. Shell Programming continued
5. Programming based on Processes and Threads
6. Processes and Threads continued
7. CPU Scheduling algorithms-FCFS & SJF
8. CPU Scheduling algorithms-RR & Priority
9. Programming based on Deadlock
10. Programming based on Deadlock continued

Unix Commands	**Description**
cd	Change directory
cd-	Return to previous directory
mkdir	Make directory
find	Find files
cat	It display file contents
pwd	To know about present working directory
ls	List all files in a directory
ls -l	List all files in long format, one file per line
ls -a	List all files including hidden files
mv	To rename the existing file
cp	To copy one or more files
man	It displays the manual pages for a chosen Unix command
rm	It removes files or directories
echo	It displays a line of text on the screen
clear	Clear the screen
who	Displays data about all the user have logged into the system currently

Exercise: Enter these commands at the UNIX prompt, and try to interpret the output:

i) echo hello world
ii) passwd
iii) date
iv) hostname
v) uname -a
vi) uptime
vii) who am i
viii) who
ix) id

x) last
xi) finger
xii) top (you may need to press q to quit)
xiii) echo $SHELL
xiv) man "automatic door"
xv) man ls (you may need to press q to quit)
xvi) man who (you may need to press q to quit)
xvii) lost
xviii) clear
xix) cal 2000
xx) bc -l (type quit or press Ctrl-d to quit)
xxi) echo 5+4 | bc -l
xxii) history

1. Write a shell script program to read two numbers and perform basic arithmetic operations (+ , - , * , / , %)

Algorithm:

```
Step 1: Start
Step 2: Read two integers a, b
Step 3: Calculate Sum= a + b
                  Diff= a – b
                  Product= a * b
                  Div=a / b
                  Rem=a % b
Step 4: Display Sum, Diff, Product, Div and Rem
Step 5: Stop
```

2. Write a shell script to read three integer numbers and print the largest among three numbers.

Algorithm:

```
Step 1: Start
Step 2: Declare variables a, b and c.
Step 3: Read variables a, b and c.
Step 4: if a>b
        if a>c
          Display a is the largest number.
        else
          Display c is the largest number.
        else
        if b>c
          Display b is the largest number.
        else
          Display c is the greatest number.
Step 5: Stop
```

3. Write a shell script program to read a character from keyboard and check whether it is vowel or not.

Algorithm:

```
Step1: Start
Step2: Declare variable ch.
Step3: Read the value of ch.
Step4: if (ch==’A’ || ch==’a’ || ch==’E’ || ch==’e’|| ch==’I’ || ch==’i’|| ch==’O’ || ch==’o’ ||
           ch==’U’ || ch == ’u’ ) then
          Display “Entered character is Vowel”
          goto Step 6
         else
Step5: Display “Entered character is not Vowel”
          goto Step 6
Step 6: Stop
```

4. Write a shell script to print out the Fibonacci series up to a limit.

Algorithm:

```
Step 1: Start
Step 2: Declare variables n, a ← 0, b ← 1, c, i
Step 3: Read values of n
Step 4: Display a, b
Step 5: Assign i←2
Step 6: if i < n then goto step 7 otherwise goto step10
Step 7: calculate c ← a+b,
                  i ← i+1
                  a ← b, b ← c
                    Display the value of c
                  goto step 6
Step 10: Stop
```

5. To write a shell script to check whether the given number is prime or not.

Algorithm:

```
Step 1: Start
Step 2: Read an integer n
Step 3: Assign i=2, j=0
Step 4: Is i < n then r =n % i. otherwise goto step 8
Step 5: Is r=0 then increment i and j value by i. otherwise go to step 6
Step 6: Increment i value by one
Step 7: Is j=0 then print number is prime and goto step 10
Step 8: Is j != 0 then print number is not a prime number
Step 9: Stop
```

6. To write a shell script to find the Armstrong numbers between 1 to N.

Algorithm:

```
Step 1: Start
Step 2: When i equal to 0 and less than or equal to N, calculate increment value of i.
Step 3: Assign value of i to temp and n.
Step 4: Assign value of ams equal to zero.
Step 5: When n not equal to zero calculate
            rem←n%10;
            ams=ams+rem*rem*rem
            n←n/10
Step 6: If temp equal to ams then print the value of ams.
Step 7: Thus for each value of i, values of ams is printed.
Step 8: Stop the program.
```

7. Write a shell script to perform Conversion of temperature in Celsius to Fahrenheit and Fahrenheit to Celsius.

Algorithm:

```
Step 1: Start
Step 2: Input the choice as 1 or 2
Step 3: Is choice is 1 then goto step 4 otherwise goto step 7
Step 4: Input temperature in Celsius
Step 5: Calculate Fahrenheit F =((9/5)*c) +32
Step 6: Print Fahrenheit F and goto step 10
Step 7: Input temperature in Fahrenheit
Step 8: Calculate Celsius C=((5/9)*(f-32))
Step 9: Print Celsius C
Step 10: Stop
```

8. Write a shell script to read an integer find out the reverse of the integer using function and check whether integer is palindrome or not.

Algorithm:

```
Step 1: Start
Step 2: read n
Step 3: copy n into m for later use. Also, initialize rn;
Step 5: while n is not zero
        1. r = n % 10
        2. n = n/10
        3. rn = rn*10 + r;
Step 6: if m equal rn then the number is palindrome.
Step 7: Else Print number is not palindrome
Step 8: Stop
```

9. Write a shell script to read an integer find out the factorial of the integer.

Algorithm:

```
Step1: Start
Step2: Read a number ‘n’ and fact=1
Step3: if n==1 then
            Return (1)
Step4: else
            For i=0 to i<n
            Factorial=fact*fact(n-1)
            Return(fact)
```

```
Step4: Stop
```

10. Write a shell script program to read an array of ‘n’ integers and perform linear search operation.

Algorithm:

```
Step 1: Start
Step 2: Read the array A of ‘n’ elements, f=0
Step 3: Read the element ‘x’ to be searched in A
Step 4: Set i to 0
Step 5: if i > n then go to step 10
Step 6: if A[i] = x then f=1 and go to step 9
Step 7: Set i to i + 1
Step 8: Go to Step 5
Step 9: Print Element x Found at index i+1 and go to step 11
Step 10: if f=0 then Print element not found
Step 11: Stop
```

11. Write a shell script program to read an array of ‘n’ integers and sort number in ascending order using bubble sort technique.

Algorithm:

```
Step1: Start
Step2: Read the number of array elements
Step3: for i = 0 to n-1
           Read array[i]
Step4: for i = 0 to n-1
           for j = 0 to n-i-1
           if (array[i]>array[j+1]) then
           Temp=array[j]
          Array[j]=array[j+1]
          Array[j+1]=temp
Step7: Display array elements
Step8: Stop
```

12. Write a shell script program to read an array of ‘n’ integers and perform binary search.

Algorithm:

```
Step 1:  Start
Step 2:  Read the array a of n elements, f=0
Step 3:  Sort using any algorithm
Step 4:  Read the element to be searched in x
Step 5:  Set L=0 the lower limit and u=n-1 the upper limit
Step 6:  Repeat the steps 7, 8, 9, 10 until u>=L
Step 7:  mid = (L+u)/2
Step 8:   when a[mid]==x f=1 print the search is successful, display the position goto step 12
Step 9:  when a[mid]<x L=mid+1
Step 10: when a[mid]>x u=mid-1
Step 11: if f==0 print search is unsuccessful
```

Step 12: Stop

SCHEDULING ALGORITHMS

1. First Come First Serve Scheduling (FCFS Scheduling)

i) Jobs are executed on first come and first serve basis
ii) It is a non pre-emptive scheduling algorithm
iii) It is easy to understand and implement
iv) Its implementation is based on first in first out (FIFO) queue
v) It is poor in performance as average waiting time is high

AIM: To write the program to implement CPU & scheduling algorithm for first come first serve scheduling.

Algorithm:

1. Start the program.
2. Get the number of processes and their burst time.
3. Initialize the waiting time for process 1 and 0.
4. Process for (i=2; i<=n; i++),wt.p[i]=p[i-1]+bt.p[i-1].
5. The waiting time of all the processes is summed then average value time is calculated.
6. The waiting time of each process and average times are displayed
7. Stop the program

2. Shortest Job First Scheduling (SJF Scheduling)

i) It is a non pre-emptive scheduling algorithm
ii) It is best approach to minimize waiting time
iii) It is easy to implement in batch systems where required CPU time is known in advance
iv) Impossible to implement in interactive systems where required CPU time is not known
v) The processor should know in advance how much time process will take

To write a program to implement cpu scheduling algorithm for shortest job first scheduling.

Algorithm:

1. Start the program. Get the number of processes and their burst time.
2. Initialize the waiting time for process 1 as 0.
3. The processes are stored according to their burst time.
4. The waiting time for the processes are calculated as follows:
   for(i=2;i<=n;i++).wt.p[i]=p[i=1]+bt.p[i-1].
5. The waiting time of all the processes summed and then the average time is calculate
6. The waiting time of each processes and average time are displayed.
7. Stop the program.

3. Priority Scheduling

i) SJF scheduling is special case of priority scheduling
ii) Priority is associated with each process
iii) CPU is allotted to the process with the highest priority

iv) For the case of equal priority, processes are scheduled on the basis of FCFS
v) It is a non pre-emptive scheduling
vi) Priority can be decided based on memory or time requirements or any other resource requirements

To write a 'C' program to perform priority scheduling.

Algorithm:
1. Start the program.
2. Read burst time, waiting time, turn the around time and priority.
3. Initialize the waiting time for process 1 and 0.
4. Based up on the priority process are arranged
5. The waiting time of all the processes is summed and then the average waiting time
6. The waiting time of each process and average waiting time are displayed based on the priority.
7. Stop the program.

4. Round Robin Scheduling

i) It is pre-emptive process scheduling algorithm
ii) Each process is provided a fix time to execute, it is called a quantum
iii) Once a process is executed for a given time period, it will be pre-empted at that given time and other process will execute for a given time period

To write a program to implement CPU scheduling for Round Robin Scheduling.

Algorithm:
1. Get the number of process and their burst time.
2. Initialize the array for Round Robin circular queue as '0'.
3. The burst time of each process is divided and the quotients are stored on the round Robin array.
4. According to the array value the waiting time for each process and the average time are calculated as line the other scheduling.
5. The waiting time for each process and average times are displayed.
6. Stop the program.

PIPE PROCESSING

To write a program for create a pipe processing

Algorithm:
1. Start the program.
2. Declare the variables.
3. Read the choice.
4. Create a piping processing using IPC.
5. Assign the variable lengths
6. "strcpy" the message lengths.
7. To join the operation using IPC.
8. Stop the program

SIMULATE ALGORITHM FOR DEADLOCK PREVENTION

Algorithm:

1. Start the program
2. Attacking Mutex condition: never grant exclusive access. But this may not be possible for several resources.
3. Attacking preemption: not something you want to do.
4. Attacking hold and wait condition: make a process hold at the most 1 resource
5. At a time. Make all the requests at the beginning. Nothing policy. If you feel, retry.
6. Attacking circular wait: Order all the resources.
7. Correct order so that there are no cycles present in the resource graph. Resources numbered 1 ... n.
8. Resources can be requested only in increasing
9. Order. i.e., you cannot request a resource whose no is less than any you may be holding.
10. Stop the program

Sd.
(M K Singh)
Professor/M&C