Operating System

CSL303

Lab Workbook

Faculty Name: Preeti Thareja

Student name: Piyush Gambhir
Roll No.: 21CSU349
Semester: V
Group: AL-3 (AIML-B)

Department of Computer Science and Engineering
The NorthCap University
Gurugram- 122017, India
Session 2023-

NAME: Piyush Gombhir
CLASS: AIML-B (AL-3)
ROLL NO: 21CSU349

S. No.	Experiment	Date of Experiment	Date of Submission	CO Covered	Signature
1	To familiarize the students to Linux interface and install Linux.	01-08-2023	16.08.2023		preeti
2	To write the shell programming code for the following. a) Write A Shell Program of Hello World b) Write a shell program to find factorial of a number. c) Write a shell program to find gross salary of an employee. d) Write a shell program to display the menu and execute instructions accordingly (i) List of files (ii) Process Status (iii) Date (iv)	08-08-2023	16.08.2023		preeti
3	To write the shell programming code for the following. a) Write a shell program to find Fibonacci series. b) Write a shell program to find largest of three numbers. c) Write a shell program to find average of N numbers	15-08-2023	16.08.2023		preeti
4	To write the shell programming code for the following. a) Write a shell program to check whether a number is even or odd b) Write a shell program to find whether a number is prime or not. c) Write a shell program to find whether a number is palindrome or not. d) Write a shell program to type number 1 to 7 and then print its corresponding day of	23-08-2023	23.08.2023		preeti
5	Implement the following CPU scheduling Algorithms. i) FCFS with Arrival time ii) FCFS without Arrival time	06.09.2023	06.09.2023		preeti

6	Implement the following CPU scheduling Algorithms. • SJF (Non-Preemptive) • SJTF (shortest remaining time first - Preemptive SJF)	13.09.2023	27.09.2023		preeti
7	Implement the priority scheduling.	19.09.2023	27.09.2023		preeti
8	Implement the Round Robin scheduling.	27.09.2023	27.09.2023		preeti
9	Write a program to implement reader/writer problem using semaphore	04.10.2023	04.10.2023		preeti
10	Write a program to implement Dining Philosopher's problem using semaphore	25.10.2023	01.10.2023		preeti
11	Write a program to implement Banker's algorithm for deadlock avoidance.	01.11.2023	01.11.2023		preeti
12	Write a program for page replacement policy using a) LRU b) FIFO c) Optimal.	08.11.2023	08.11.2023		preeti

Experiment No: 1

Student Name and Roll Number: Piyush Gambhir
Semester /Section: Semester V – AIML-B (AL-3)
Link to Code: NCU-Lab-Manual-And-End-Semester-Projects/NCU-CSL303-OS-Lab_Manual at main · Piyush-Gambhir/NCU-Lab-Manual-And-End-Semester-Projects (github.com)
Date: 01.08.2023
Faculty Signature:
Marks:

Objective(s):
To familiarize the students to Linux interface.

Outcome:

- The students will understand commands used in Linux.

Problem Statement:

Implement the following things:

- Linux Installation

Background Study:

Cygwin is a open source tool which provides that functionality of the Linux in windows Operating System. Cygwin is a large collection of GNU and Open Source tools which provide functionality similar to a Linux distribution on Windows. It is a DLL (cygwin1.dll) which provides substantial POSIX API functionality.

Question Bank:

Question Bank:

1. **What is Linux?**
   Linux is an open-source operating system kernel that serves as the core component of various Linux-based operating systems. It was originally developed by Linus Torvalds in 1991 and has since grown into a robust and widely used operating system. Linux is known for its stability, security, flexibility, and customizability. It is used in a wide range of devices, from servers and supercomputers to smartphones and embedded systems.

2. **How will you List files from a directory?**
   To list files from a directory in a Linux terminal, you can use the ls command. The basic syntax for listing files is:

ls [options] [directory]

3. **How files in a directory can be removed?**
   To remove files in a directory on Linux, you can use the rm command, which stands for "remove." Be cautious when using this command, as it will permanently delete files, and there is no direct way to recover them. The basic syntax is:
   rm [options] file1 file2 ...

4. **How to find out a word in a file?**
   To find a specific word in a file on Linux, you can use the grep command. The grep command is used to search for text patterns within files. The basic syntax is:
   grep "word" filename

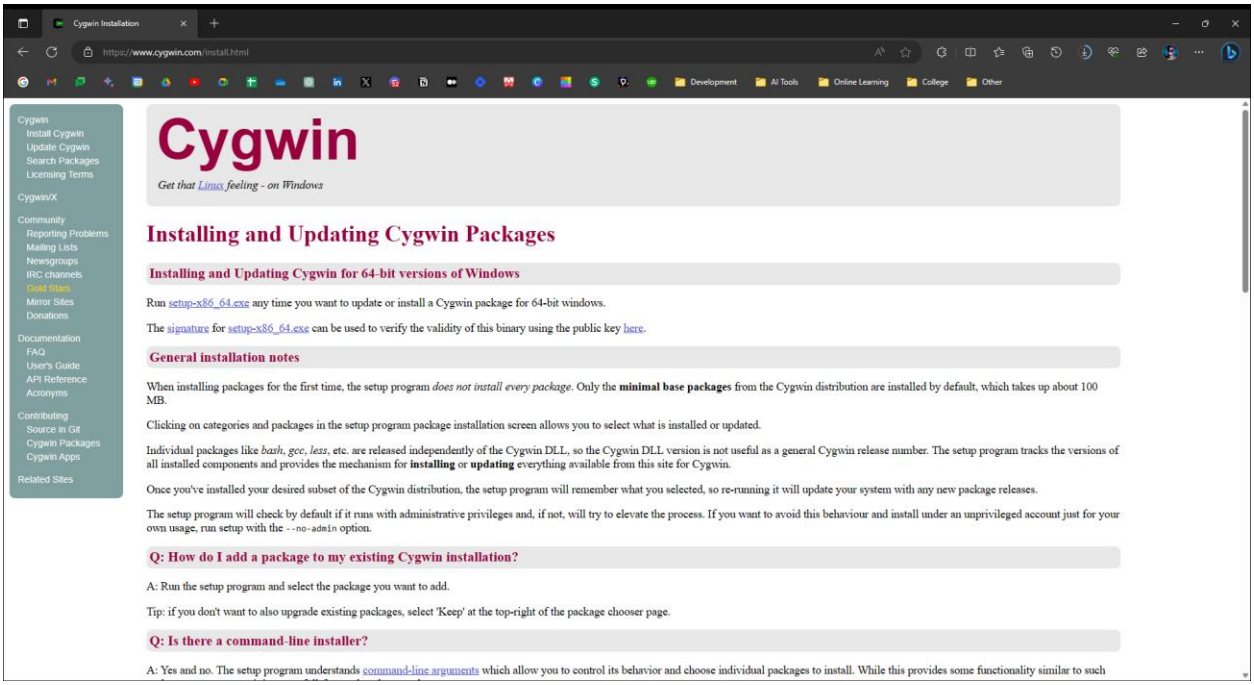
   Here, "word" is the word you want to search for, and filename is the name of the file in which you want to search for the word.

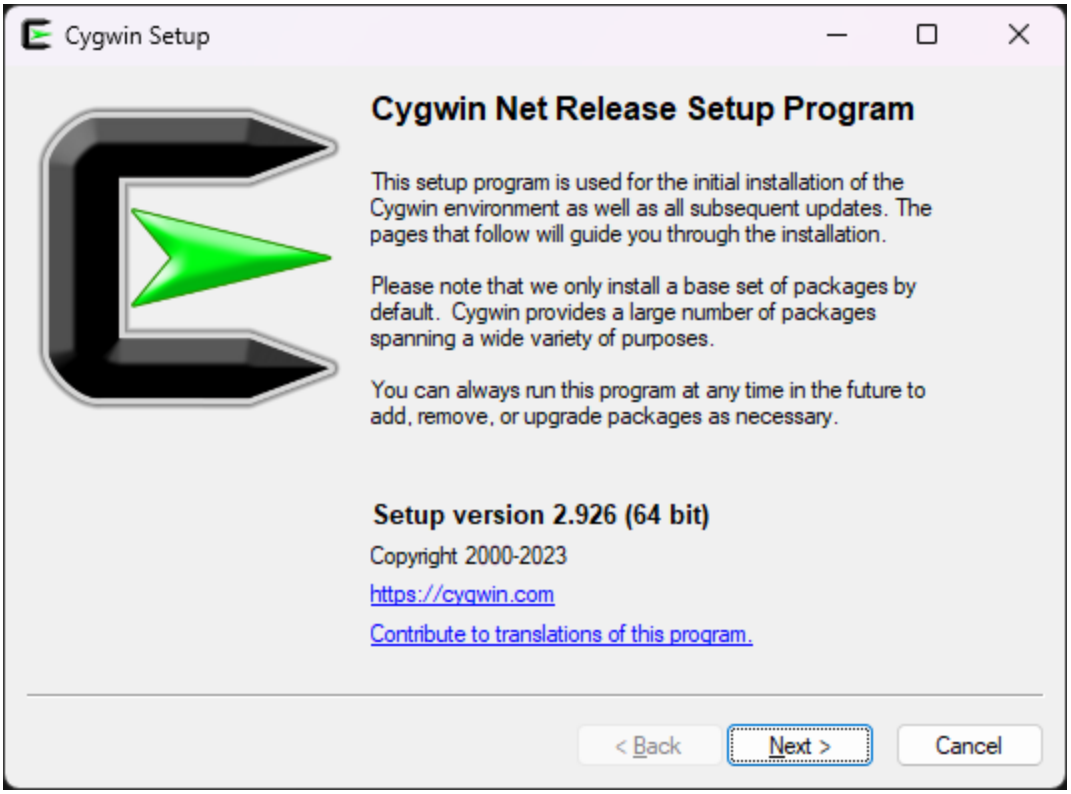
5. **What are wildcards?**
   Wildcards are special characters used in Linux shell commands to represent patterns of filenames or directories. They allow you to perform actions on multiple files or directories that match a certain pattern. Common wildcards include * (asterisk) for representing zero or more characters, ? (question mark) for representing a single character, [ ] (square brackets) for representing a range of characters, and { } (curly braces) for grouping patterns together. Wildcards are especially useful when combined with commands like ls, cp, mv, and rm to perform actions on multiple files at once based on specific patterns.

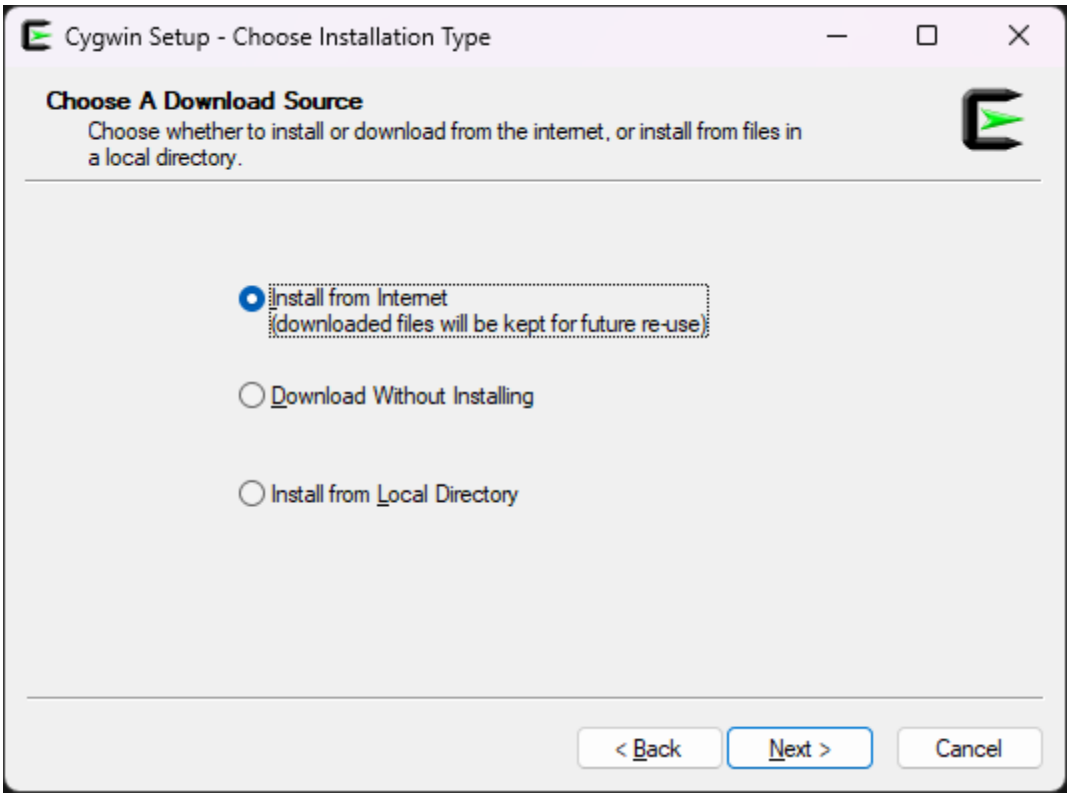
Student Work Area

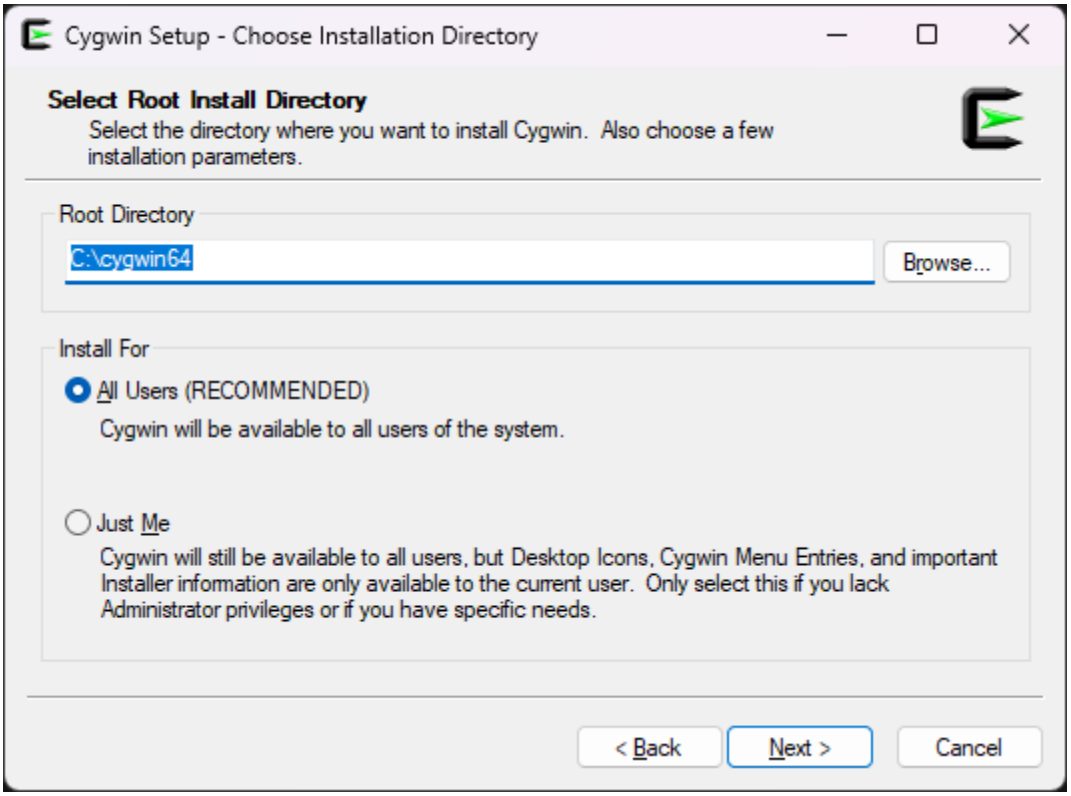
Algorithm/Flowchart/Code/Sample Outputs

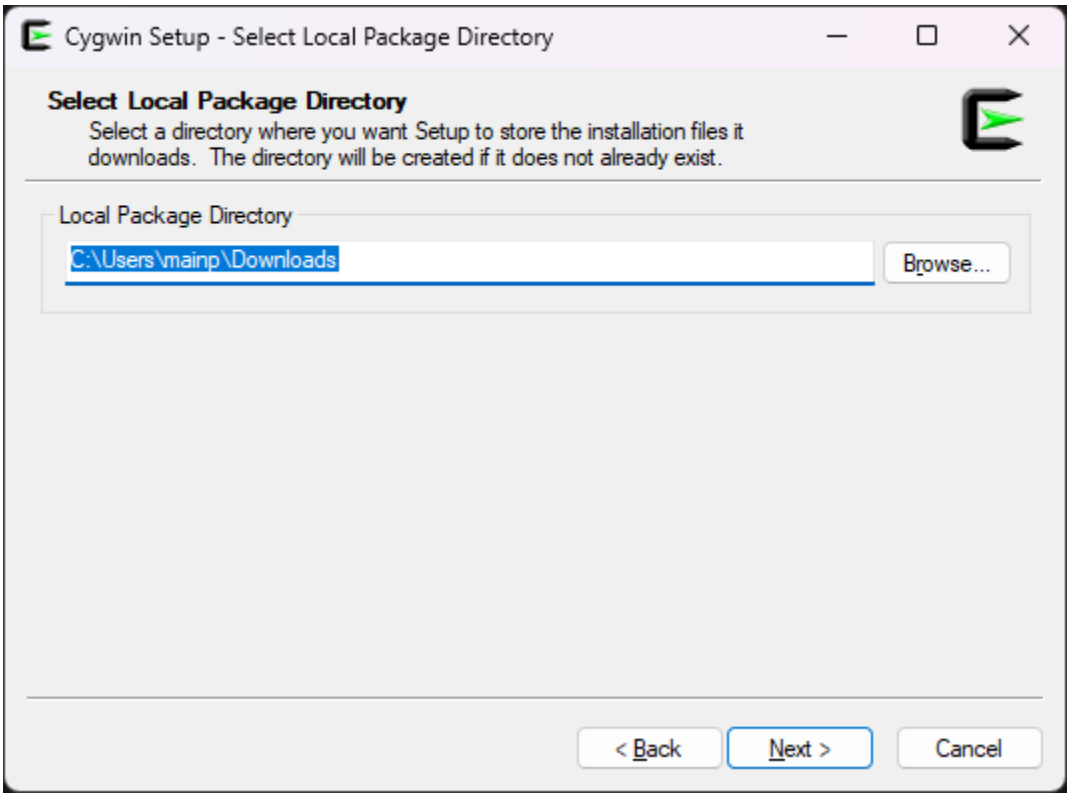
INSTALLATION

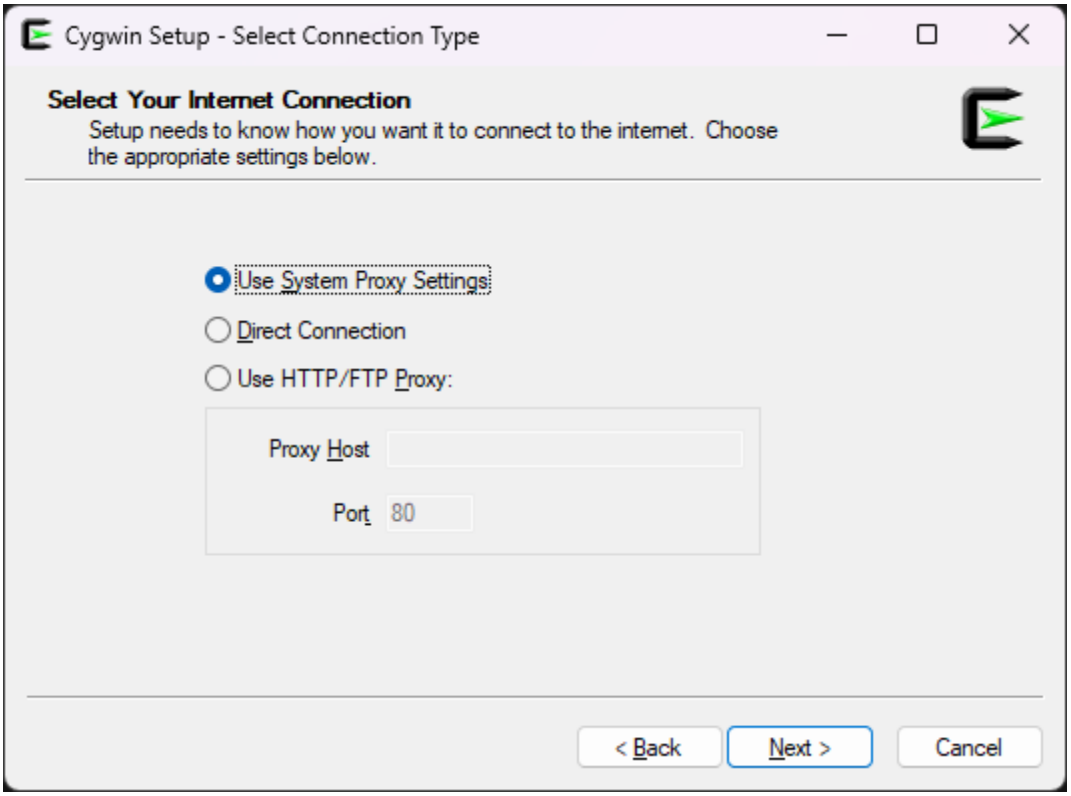


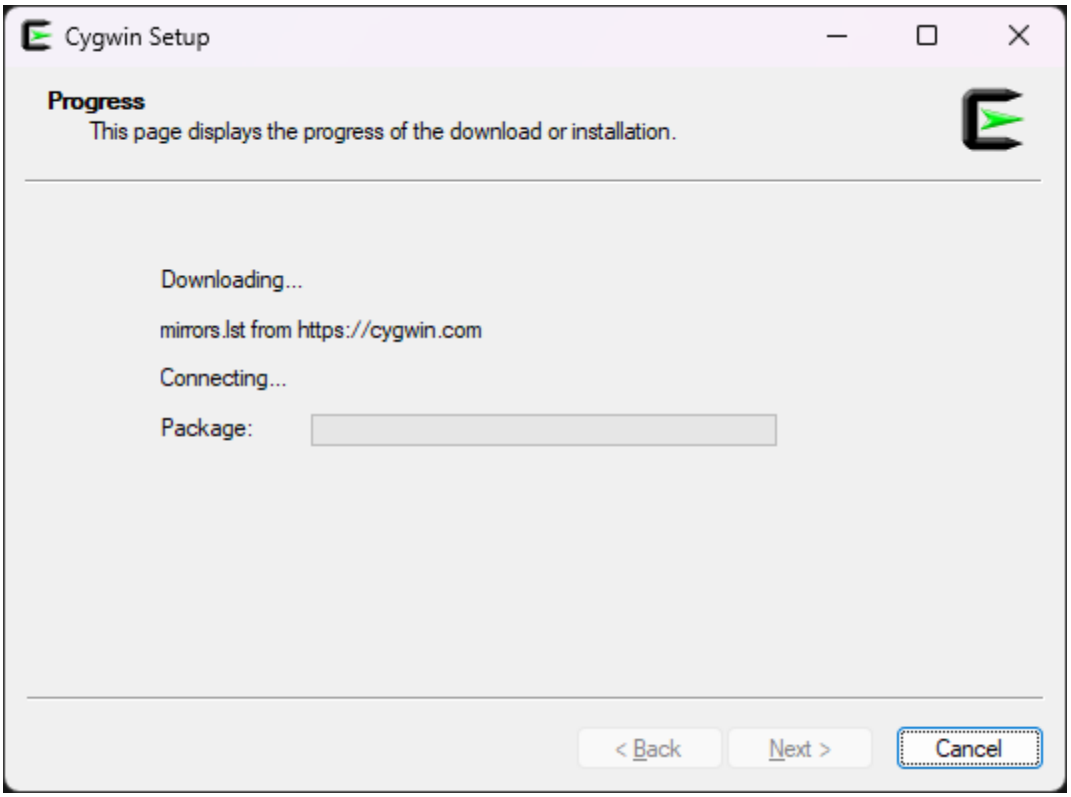

Cygwin Setup
Cygwin Net Release Setup Program
This setup program is used for the initial installation of the Cygwin environment as well as all subsequent updates. The pages that follow will guide you through the installation.
Please note that we only install a base set of packages by default. Cygwin provides a large number of packages spanning a wide variety of purposes.
You can always run this program at any time in the future to add, remove, or upgrade packages as necessary.
Setup version 2.926 (64 bit)
Copyright 2000-2023
https://cygwin.com
Contribute to translations of this program.
< Back
Next >
Cancel

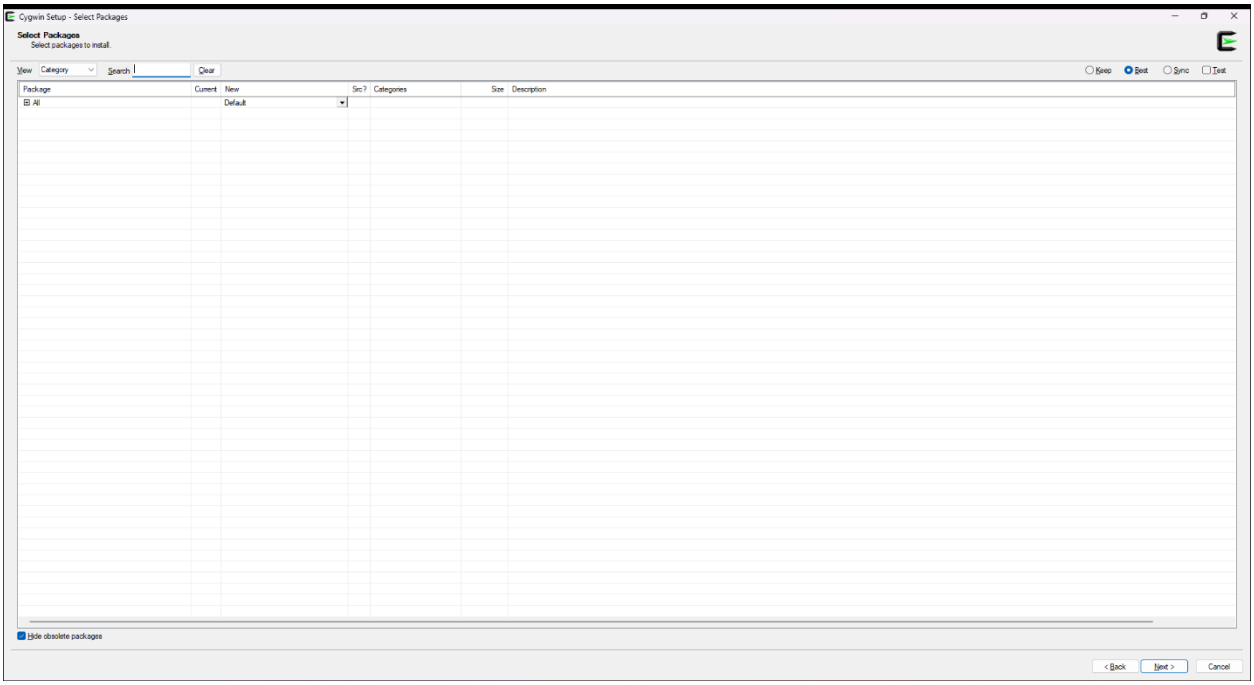

Cygwin Setup - Choose Installation Type
Choose A Download Source
Choose whether to install or download from the internet, or install from files in a local directory.
Install from Internet
(downloaded files will be kept for future re-use)
Download Without Installing
Install from Local Directory
< Back
Next >
Cancel

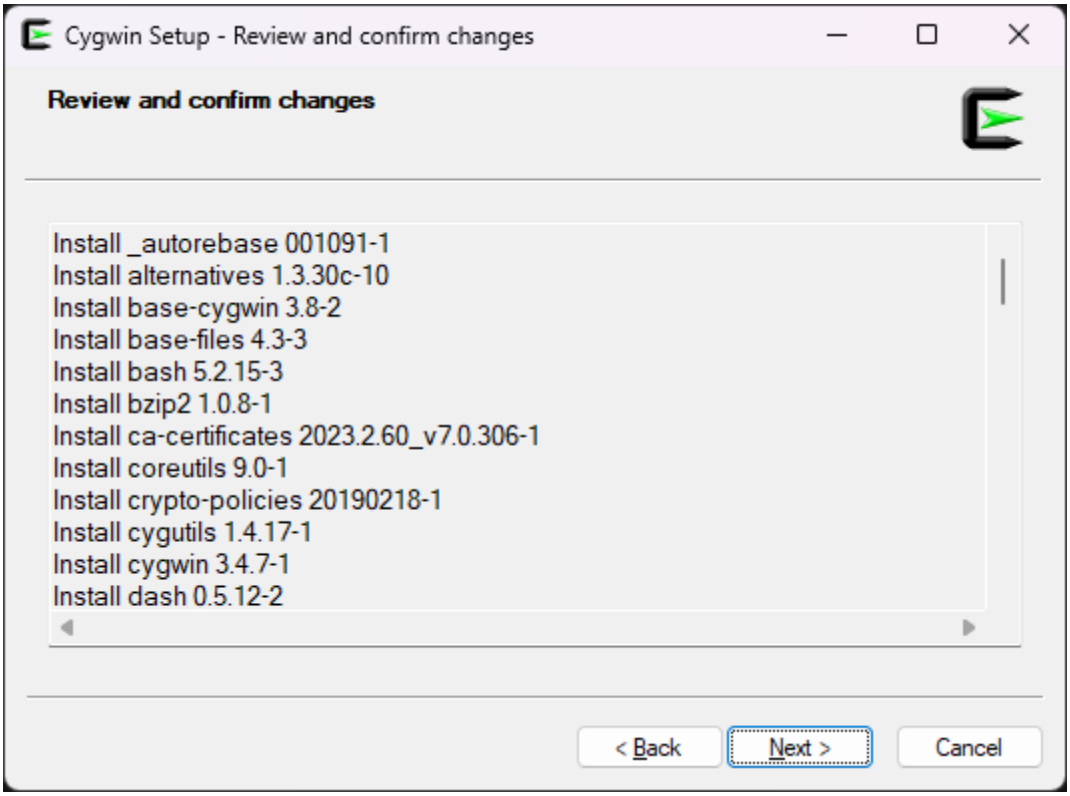

Cygwin Setup - Choose Installation Directory
Select Root Install Directory
Select the directory where you want to install Cygwin. Also choose a few installation parameters.
Root Directory
C:\cygwin64
Browse...
Install For
All Users (RECOMMENDED)
Cygwin will be available to all users of the system.
Just Me
Cygwin will still be available to all users, but Desktop Icons, Cygwin Menu Entries, and important Installer information are only available to the current user. Only select this if you lack Administrator privileges or if you have specific needs.
< Back
Next >
Cancel

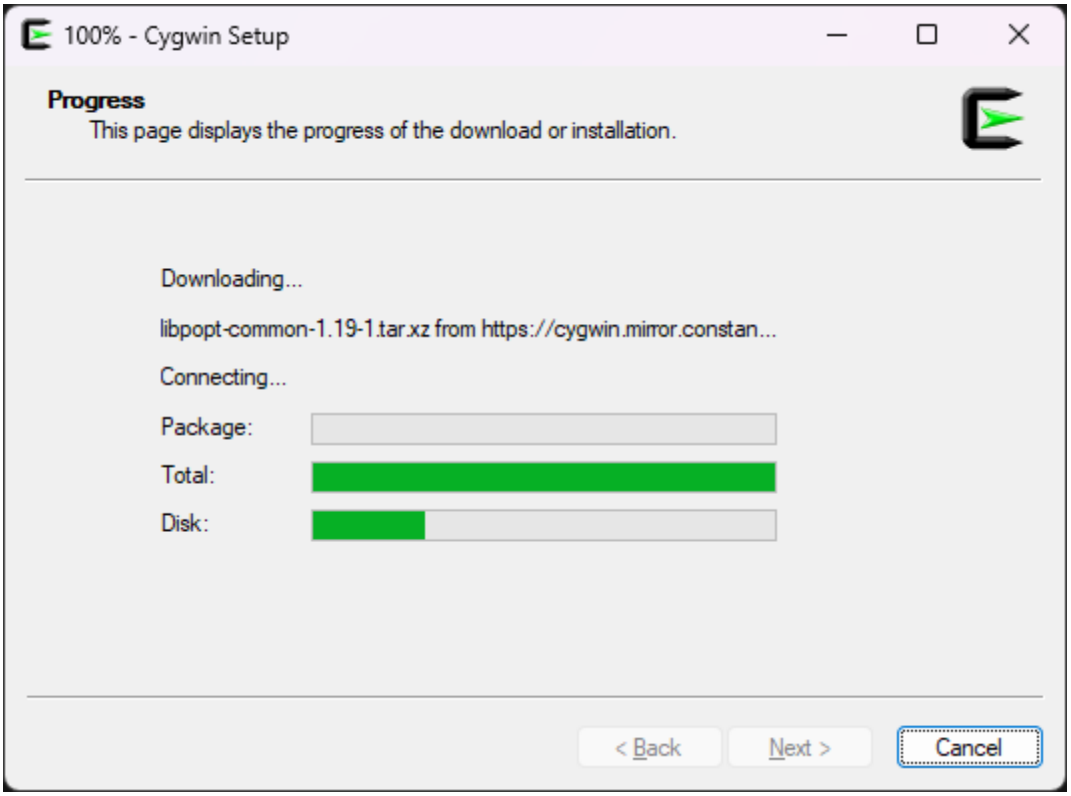

Cygwin Setup - Select Local Package Directory
Select Local Package Directory
Select a directory where you want Setup to store the installation files it downloads. The directory will be created if it does not already exist.
Local Package Directory
C:\Users\mainp\Downloads
Browse...
< Back
Next >
Cancel

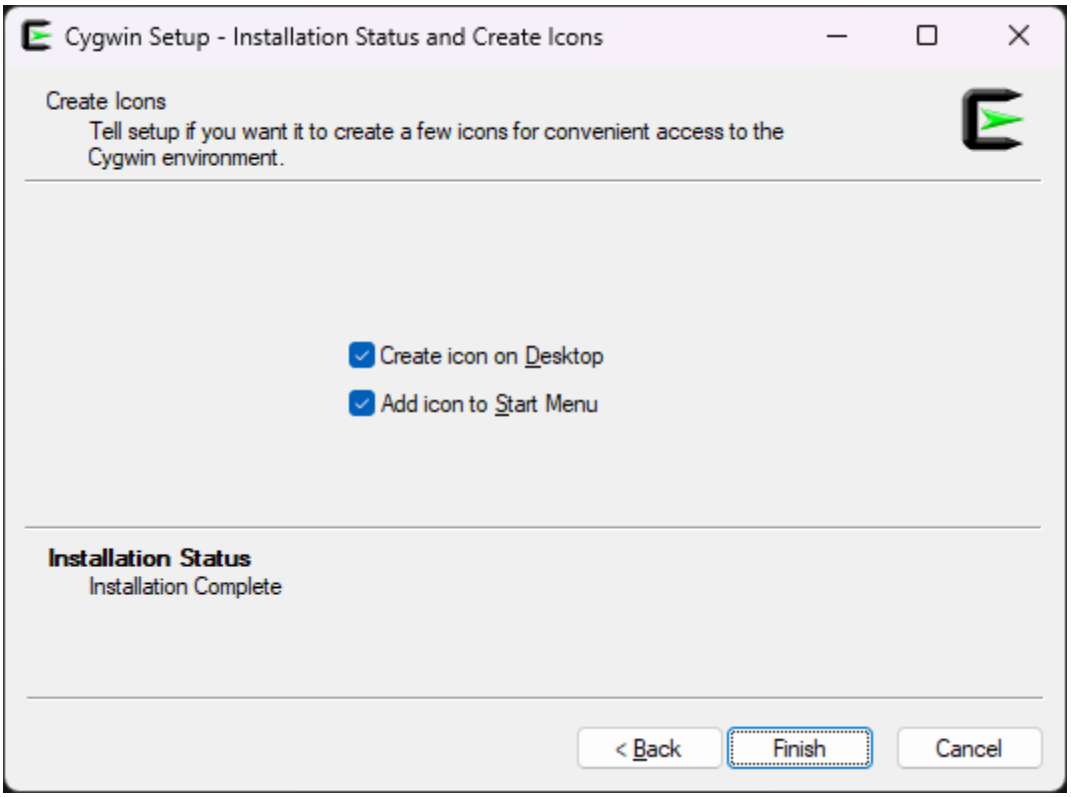

Cygwin Setup - Select Connection Type
Select Your Internet Connection
Setup needs to know how you want it to connect to the internet. Choose the appropriate settings below.
Use System Proxy Settings
Direct Connection
Use HTTP/FTP Proxy:
Proxy Host
Port 80
< Back
Next >
Cancel

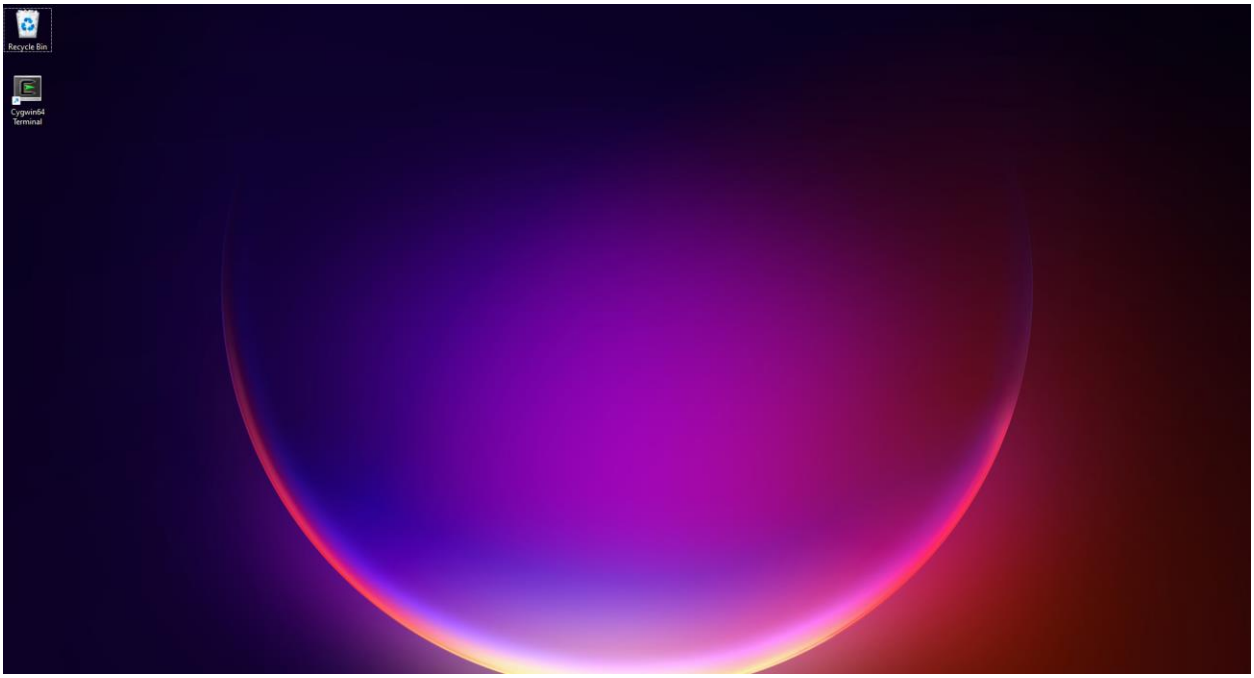

Cygwin Setup
Progress
This page displays the progress of the download or installation.
Downloading...
mirrors.lst from https://cygwin.com
Connecting...
Package:
< Back
Next >
Cancel

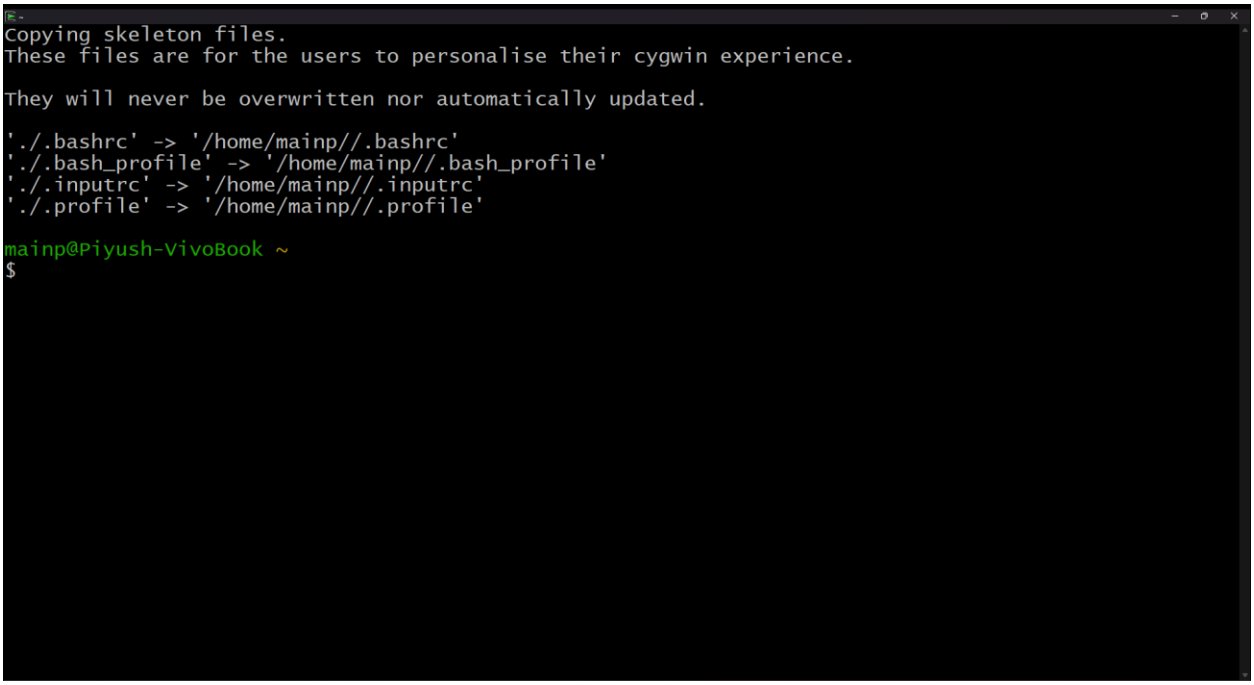

Cygwin Setup - Select Packages
Select Packages
Select packages to install.
View
Category
Search
Clear
Keep
Prev
Sync
Test
Package
Current
New
Src?
Categories
Size
Description
All
Default
Hide obsolete packages
< Back
Next >
Cancel


Cygwin Setup - Review and confirm changes
Review and confirm changes
Install _autorebase 001091-1
Install alternatives 1.3.30c-10
Install base-cygwin 3.8-2
Install base-files 4.3-3
Install bash 5.2.15-3
Install bzip2 1.0.8-1
Install ca-certificates 2023.2.60_v7.0.306-1
Install coreutils 9.0-1
Install crypto-policies 20190218-1
Install cygutils 1.4.17-1
Install cygwin 3.4.7-1
Install dash 0.5.12-2
< Back
Next >
Cancel

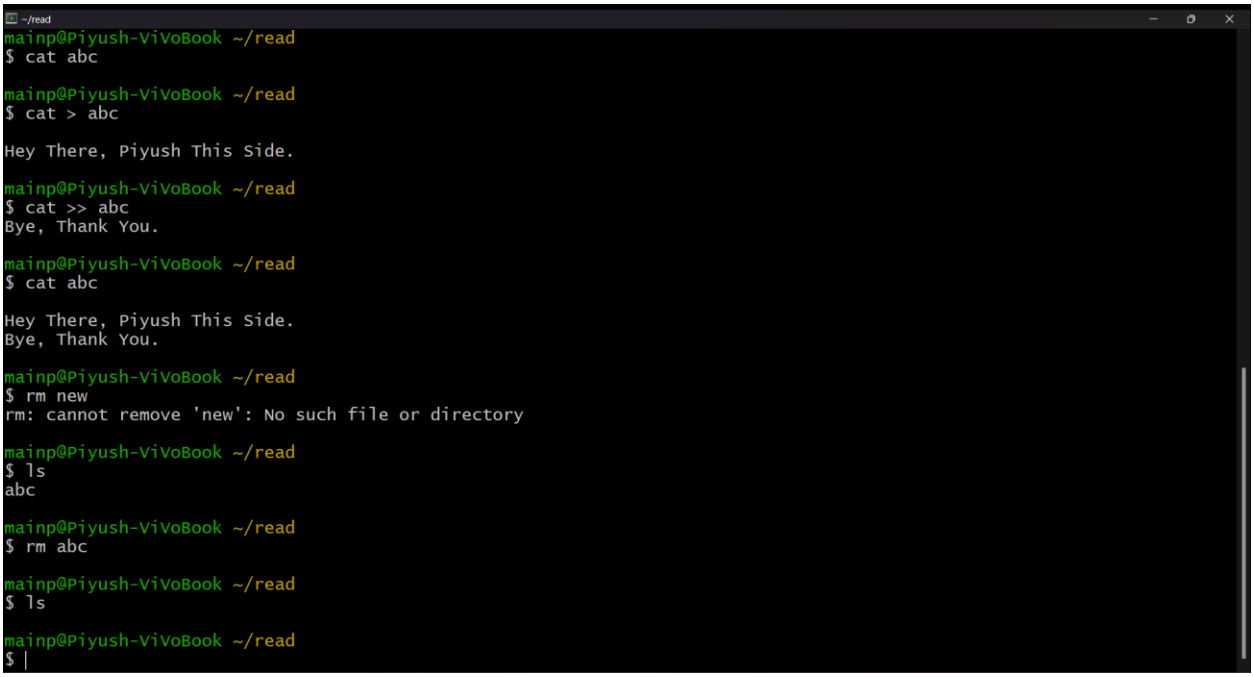

100% - Cygwin Setup
Progress
This page displays the progress of the download or installation.
Downloading...
libpopt-common-1.19-1.tar.xz from https://cygwin.mirror.constan...
Connecting...
Package:
Total:
Disk:
< Back
Next >
Cancel

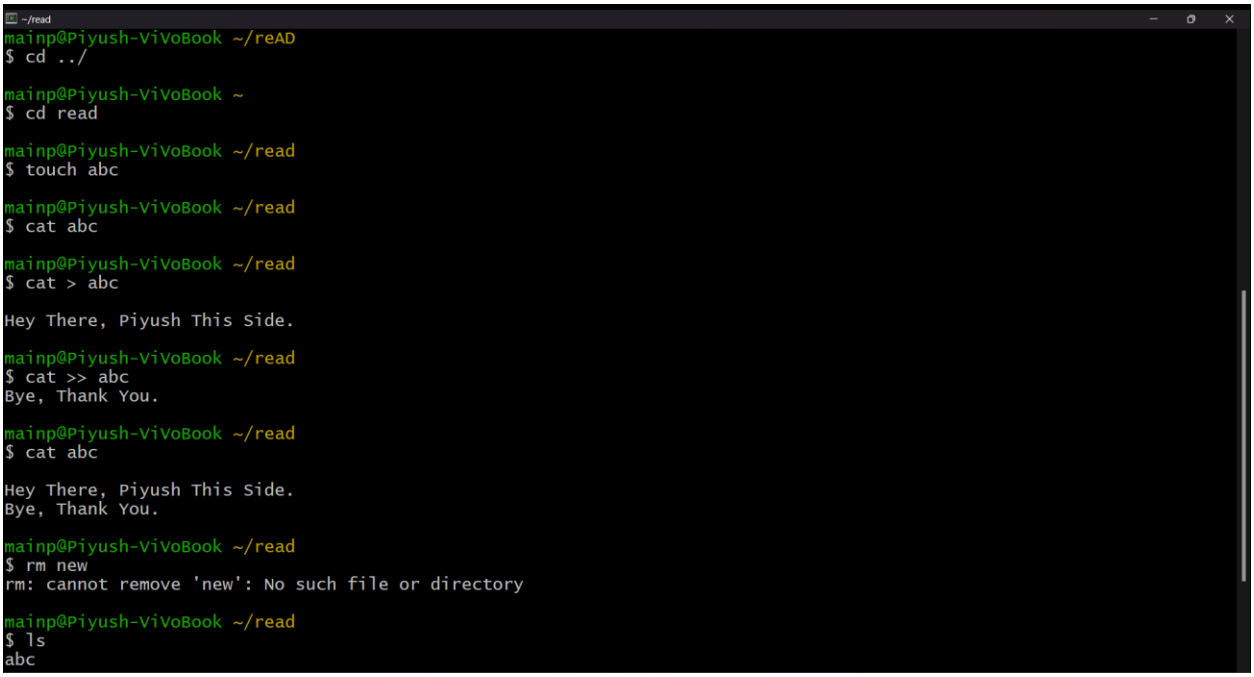

Cygwin Setup - Installation Status and Create Icons
Create Icons
Tell setup if you want it to create a few icons for convenient access to the Cygwin environment.
Create icon on Desktop
Add icon to Start Menu
Installation Status
Installation Complete
< Back
Finish
Cancel

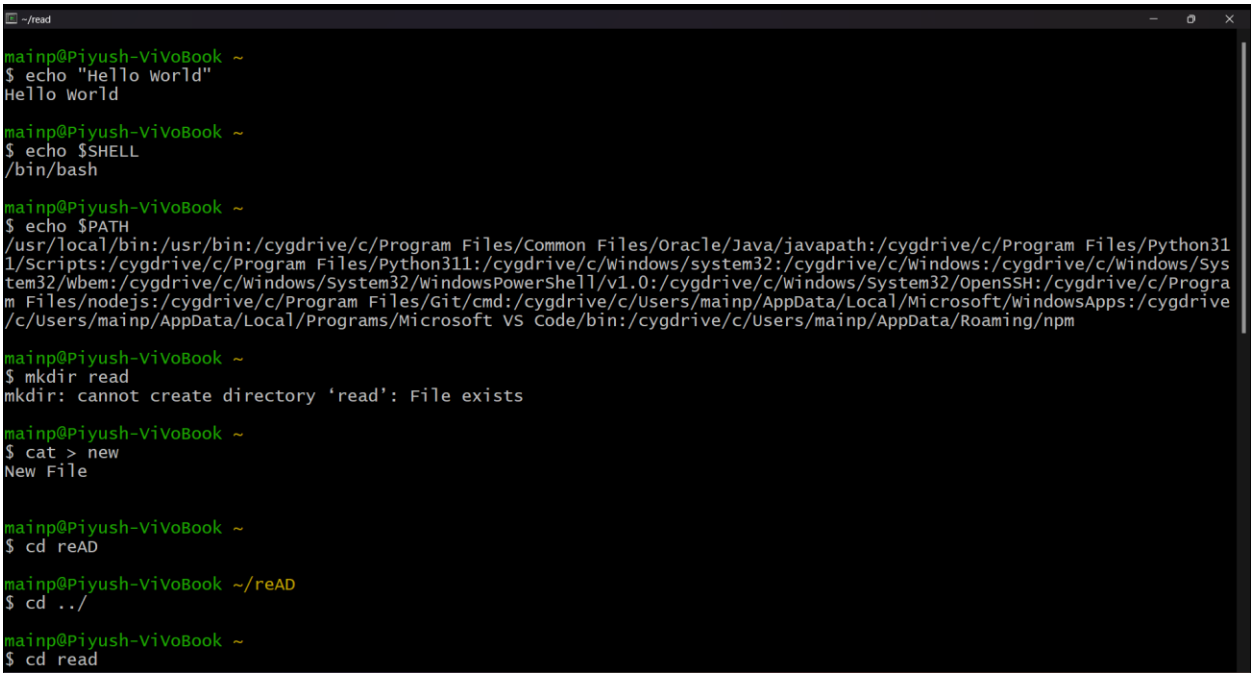

Recycle Bin
Cygwin64 Terminal

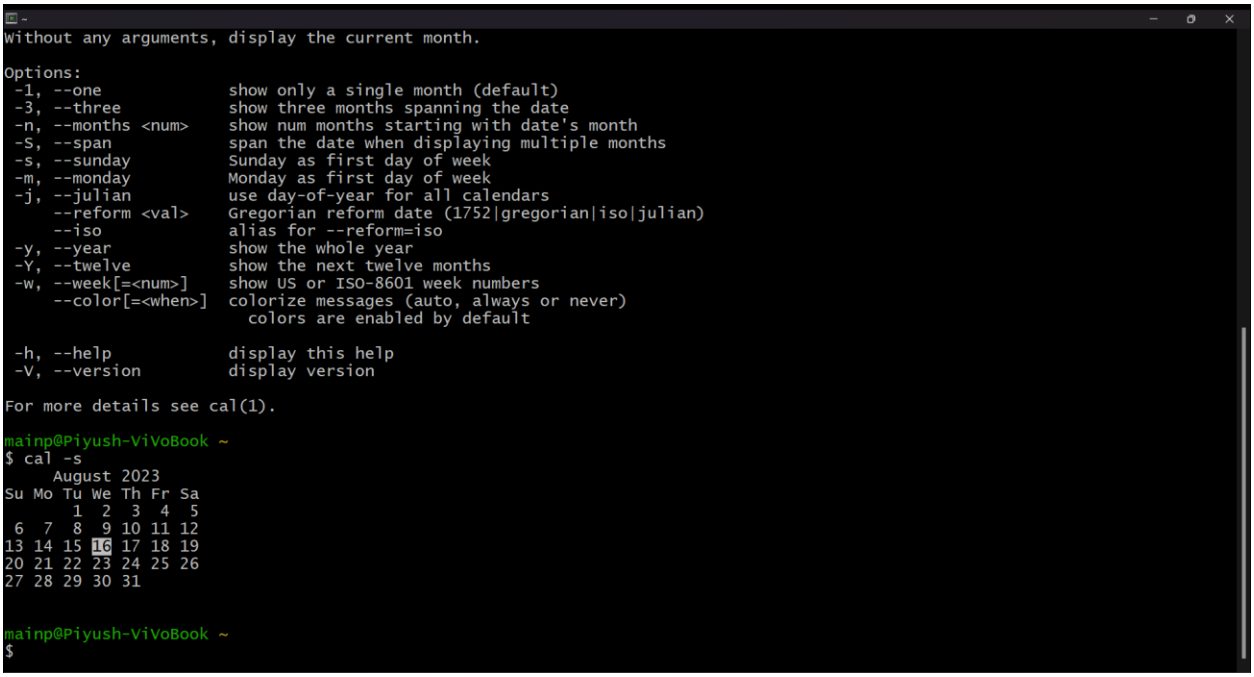

Copying skeleton files.
These files are for the users to personalise their cygwin experience.
They will never be overwritten nor automatically updated.
'./.bashrc' -> '/home/mainp//.bashrc'
'./.bash_profile' -> '/home/mainp//.bash_profile'
'./.inputrc' -> '/home/mainp//.inputrc'
'./.profile' -> '/home/mainp//.profile'
mainp@Piyush-VivoBook ~
$

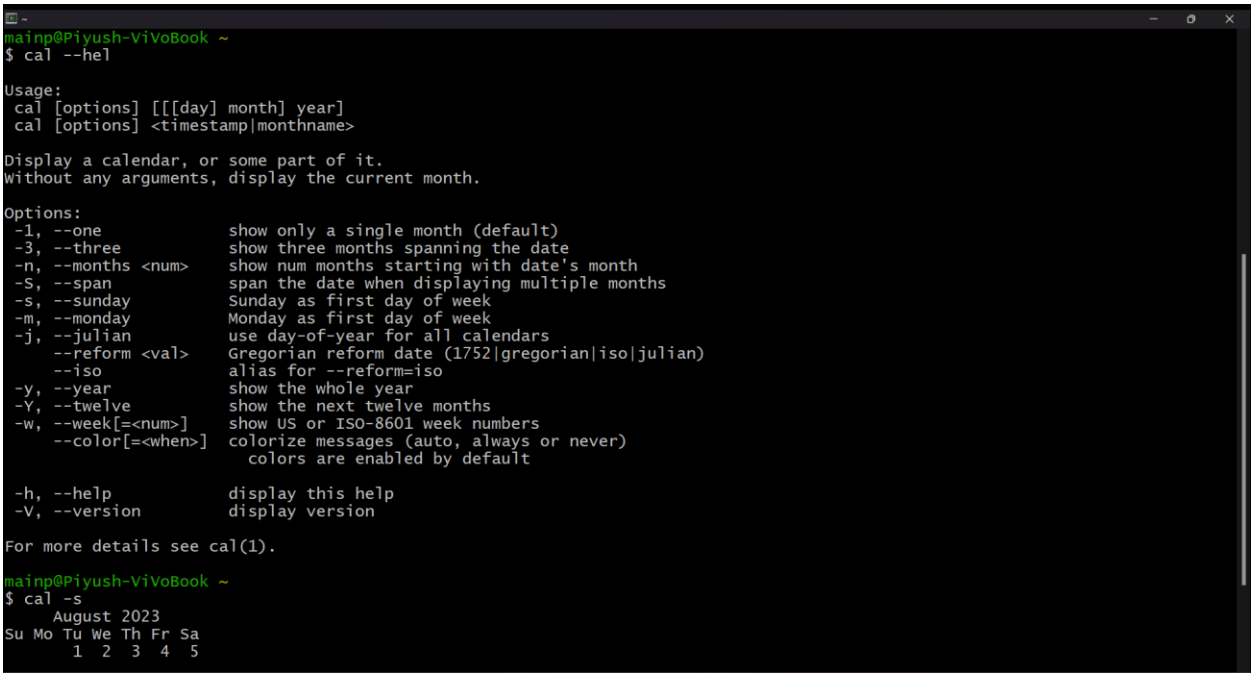
BASIC COMMANDS

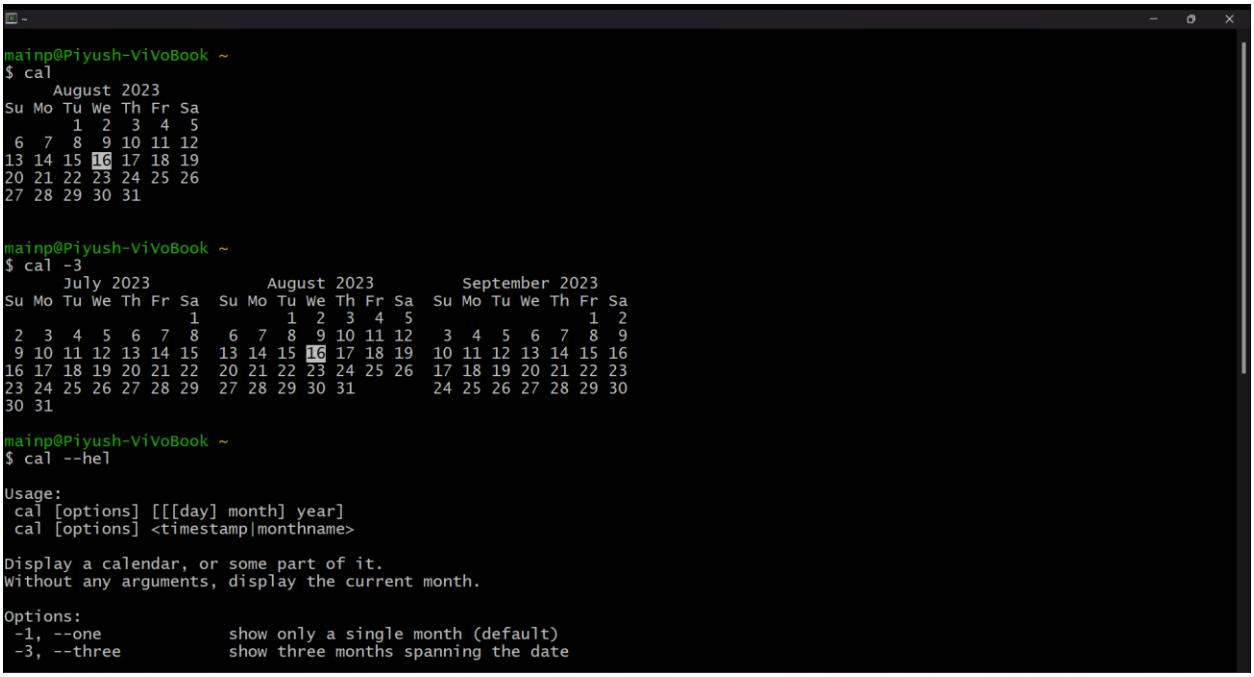



~/read
mainp@Piyush-ViVoBook ~/reAD
$ cd ../
mainp@Piyush-ViVoBook ~
$ cd read
mainp@Piyush-ViVoBook ~/read
$ touch abc
mainp@Piyush-ViVoBook ~/read
$ cat abc
mainp@Piyush-ViVoBook ~/read
$ cat > abc
Hey There, Piyush This Side.
mainp@Piyush-ViVoBook ~/read
$ cat >> abc
Bye, Thank You.
mainp@Piyush-ViVoBook ~/read
$ cat abc
Hey There, Piyush This Side.
Bye, Thank You.
mainp@Piyush-ViVoBook ~/read
$ rm new
rm: cannot remove 'new': No such file or directory
mainp@Piyush-ViVoBook ~/read
$ ls
abc


~/read
mainp@Piyush-ViVoBook ~
$ echo "Hello World"
Hello World
mainp@Piyush-ViVoBook ~
$ echo $SHELL
/bin/bash
mainp@Piyush-ViVoBook ~
$ echo $PATH
/usr/local/bin:/usr/bin:/cygdrive/c/Program Files/Common Files/Oracle/Java/javapath:/cygdrive/c/Program Files/Python311/Scripts:/cygdrive/c/Program Files/Python311:/cygdrive/c/Windows/system32:/cygdrive/c/Windows:/cygdrive/c/Windows/System32/Wbem:/cygdrive/c/Windows/System32/WindowsPowerShell/v1.0:/cygdrive/c/Windows/System32/OpenSSH:/cygdrive/c/Program Files/nodejs:/cygdrive/c/Program Files/Git/cmd:/cygdrive/c/Users/mainp/AppData/Local/Microsoft/WindowsApps:/cygdrive/c/Users/mainp/AppData/Local/Programs/Microsoft VS Code/bin:/cygdrive/c/Users/mainp/AppData/Roaming/npm
mainp@Piyush-ViVoBook ~
$ mkdir read
mkdir: cannot create directory ‘read’: File exists
mainp@Piyush-ViVoBook ~
$ cat > new
New File
mainp@Piyush-ViVoBook ~
$ cd reAD
mainp@Piyush-ViVoBook ~/reAD
$ cd ../
mainp@Piyush-ViVoBook ~
$ cd read


Without any arguments, display the current month.
Options:
-1, --one show only a single month (default)
-3, --three show three months spanning the date
-n, --months <num> show num months starting with date's month
-S, --span span the date when displaying multiple months
-s, --sunday Sunday as first day of week
-m, --monday Monday as first day of week
-j, --julian use day-of-year for all calendars
--reform <val> Gregorian reform date (1752|gregorian|iso|julian)
--iso alias for --reform=iso
-y, --year show the whole year
-Y, --twelve show the next twelve months
-w, --week[=<num>] show US or ISO-8601 week numbers
--color[=<when>] colorize messages (auto, always or never)
colors are enabled by default
-h, --help display this help
-V, --version display version
For more details see cal(1).
mainp@Piyush-ViVoBook ~
$ cal -s
August 2023
Su Mo Tu We Th Fr Sa
1 2 3 4 5
6 7 8 9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31
mainp@Piyush-ViVoBook ~
$


mainp@Piyush-ViVoBook ~
$ cal --hel
Usage:
cal [options] [[[day] month] year]
cal [options] <timestamp|monthname>
Display a calendar, or some part of it.
Without any arguments, display the current month.
Options:
-1, --one show only a single month (default)
-3, --three show three months spanning the date
-n, --months <num> show num months starting with date's month
-S, --span span the date when displaying multiple months
-s, --sunday Sunday as first day of week
-m, --monday Monday as first day of week
-j, --julian use day-of-year for all calendars
--reform <val> Gregorian reform date (1752|gregorian|iso|julian)
--iso alias for --reform=iso
-y, --year show the whole year
-Y, --twelve show the next twelve months
-w, --week[=<num>] show US or ISO-8601 week numbers
--color[=<when>] colorize messages (auto, always or never)
colors are enabled by default
-h, --help display this help
-V, --version display version
For more details see cal(1).
mainp@Piyush-ViVoBook ~
$ cal -s
August 2023
Su Mo Tu We Th Fr Sa
1 2 3 4 5


mainp@Piyush-ViVoBook ~
$ cal
August 2023
Su Mo Tu We Th Fr Sa
1 2 3 4 5
6 7 8 9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31
mainp@Piyush-ViVoBook ~
$ cal -3
July 2023 August 2023 September 2023
Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa Su Mo Tu We Th Fr Sa
1 1 2 3 4 5 1 2
2 3 4 5 6 7 8 6 7 8 9 10 11 12 3 4 5 6 7 8 9
9 10 11 12 13 14 15 13 14 15 16 17 18 19 10 11 12 13 14 15 16
16 17 18 19 20 21 22 20 21 22 23 24 25 26 17 18 19 20 21 22 23
23 24 25 26 27 28 29 27 28 29 30 31 24 25 26 27 28 29 30
30 31
mainp@Piyush-ViVoBook ~
$ cal --hel
Usage:
cal [options] [[[day] month] year]
cal [options] <timestamp|monthname>
Display a calendar, or some part of it.
Without any arguments, display the current month.
Options:
-1, --one show only a single month (default)
-3, --three show three months spanning the date

Experiment No: 2

Student Name and Roll Number: Piyush Gambhir
Semester /Section: Semester V – AIML-B (AL-3)
Link to Code: NCU-Lab-Manual-And-End-Semester-Projects/NCU-CSL303-OS-Lab_Manual at main · Piyush-Gambhir/NCU-Lab-Manual-And-End-Semester-Projects (github.com)
Date: 08.08.2023
Faculty Signature:
Marks:

Objective(s):
To write the shell programming code for the following.

Outcome:

- Student is able to write code in shell programming.

Problem Statement:

a) Write A Shell Program of Hello World
b) Write a shell program to find factorial of a number.
c) Write a shell program to find gross salary of an employee.
d) Write a shell program to display the menu and execute instructions accordingly

- (i)List of files (ii)Process Status (iii) Date (iv) users in program (v) Quit

Background Study:

A shell script is a file with a set of commands in it. The shell reads this file and executes the instructions as if they were input directly on the command line.

A shell is a command-line interpreter and operations such as file manipulation, program execution and text printing are performed by shell script. So, we will use vi editor to edit our files.

Question Bank:

1. What is a shell?
   A shell is a command-line interface that acts as an intermediary between a user and the operating system. It allows users to interact with the computer by typing commands and receiving responses. The shell interprets the commands and executes them, facilitating tasks such as file manipulation, process management, and more.

2. What is the significance of $#?

$# is a special variable in shell scripting that holds the number of arguments passed to a script or a function. It's often used to determine the count of arguments provided, helping scripts adapt their behavior based on the provided input.

3. What are the different types of commonly used shells on a typical Linux system?
   Commonly used shells on Linux systems include:
   - Bash (Bourne Again Shell): Widely used default shell, known for its scripting capabilities.
   - Zsh (Z Shell): Offers advanced features and customization options.
   - Fish: Known for its user-friendly auto-suggestions and syntax highlighting.
   - Dash: A lightweight shell used for scripting, commonly used as /bin/sh.
   - Tcsh: Enhanced version of the C shell (csh) with more features.

4. How will you pass and access arguments to a script in Linux?
   Arguments can be passed to a script in Linux by providing them as space-separated values after the script's name when invoking it. Inside the script, these arguments can be accessed using positional parameters like $1 for the first argument, $2 for the second, and so on. The special variable $0 holds the script's name.

5. Use sed command to replace the content of the file (emulate tac command)

   To emulate the tac command's behavior (which reverses the lines of a file), you can use the sed command like this:

   sed '1!G;h;$!d' inputfile > outputfile

   This sed command appends each line to the hold space, then exchanges the pattern space (current line) with the hold space, effectively reversing the order of lines. The 1!G command appends the hold space to the pattern space for all lines except the first, h swaps the hold and pattern spaces, and $!d deletes all lines except the last, effectively reversing the content of the file.

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Problem Statement a:

Code:

```
: '
Problem Statement:
a) Write A Shell Program of Hello World.
'

#!/bin/bash

echo "Hello, World!"
```

Output:

Problem Statement b:

Code:

```
: '
Problem Statement:
b) Write a shell program to find factorial of a number.
'

#!/bin/bash

factorial() {
    if [ $1 -le 1 ]; then
        echo 1
    else
        last=$(factorial $(( $1 - 1 )))
        echo $(( $1 * last ))
    fi
}

read -p "Enter a number: " num
result=$(factorial $num)
echo "Factorial of $num is: $result"
```

Output:


piyushgambhir@Piyush-Note
piyushgambhir@Piyush-Notebook:/mnt/c/Users/mainp/Downloads/Experiment 2$ ./problem_statement_b.bash
Enter a number: 5
Factorial of 5 is: 120
piyushgambhir@Piyush-Notebook:/mnt/c/Users/mainp/Downloads/Experiment 2$

Problem Statement c:

Code:

```
: '
Problem Statement:
c) Write a shell program to find gross salary of an employee.
'

#!/bin/bash

read -p "Enter Basic Salary: " basic

hra=$(echo "0.20 * $basic" | bc)
da=$(echo "0.30 * $basic" | bc)

gross=$(echo "$basic + $hra + $da" | bc)

echo "Gross Salary is: $gross"
```

Output:

Problem Statement d:

Code:

```
: '
d) Problem Statement : Write a shell program to display the menu and execute
instructions accordingly:
    (i)List of files (ii)Process Status (iii) Date (iv) users in program (v) Quit
'

#!/bin/bash

while true; do
    echo "Choose an option:"
    echo "(i) List of files"
    echo "(ii) Process Status"
    echo "(iii) Date"
    echo "(iv) Users in program"
    echo "(v) Quit"

    read choice

    case $choice in
        i) ls ;;
        ii) ps ;;
        iii) date ;;
        iv) who ;;
        v) exit 0 ;;
        *) echo "Invalid choice";;
    esac
done
```

Output:


piyushgambhir@Piyush-Note
piyushgambhir@Piyush-Notebook:/mnt/c/Users/mainp/Downloads/Experiment 2$ ./problem_statement_d.bash
Choose an option:
(i) List of files
(ii) Process Status
(iii) Date
(iv) Users in program
(v) Quit
i
EXP2.pptx problem_statement_a.bash problem_statement_c.bash
'Experiment No 2-1.docx' problem_statement_b.bash problem_statement_d.bash
Choose an option:
(i) List of files
(ii) Process Status
(iii) Date
(iv) Users in program
(v) Quit
iv
piyushgambhir pts/1 2023-10-15 19:29
Choose an option:
(i) List of files
(ii) Process Status
(iii) Date
(iv) Users in program
(v) Quit
v
piyushgambhir@Piyush-Notebook:/mnt/c/Users/mainp/Downloads/Experiment 2$

EXPERIMENT NO. 3

Student Name and Roll Number: Piyush Gambhir
Semester /Section: Semester V – AIML-B (AL-3)
Link to Code: NCU-Lab-Manual-And-End-Semester-Projects/NCU-CSL303-OS-Lab_Manual at main · Piyush-Gambhir/NCU-Lab-Manual-And-End-Semester-Projects (github.com)
Date: 16.08.2023
Faculty Signature:
Marks:

Objective:
To write the shell programming code for the following.

Outcome:
Student is able to write code in shell programming

Problem Statement:
a) Write a shell program to find Fibonacci series.
b) Write a shell program to find largest of three numbers.
c) Write a shell program to find average of N numbers

Background Study:

A shell script is a file with a set of commands in it. The shell reads this file and executes the instructions as if they were input directly on the command line.

A shell is a command-line interpreter and operations such as file manipulation, program execution and text printing are performed by shell script. So, we will use vi editor to edit our files.

Question Bank:

1. How to use multi line comments in shell script?
   Use multi-line comments in a shell script using : ' ... '. For example:
   ```
   : '
   This is a
   multi-line comment.
   '
   ```
2. What is the difference between soft and hard links?
   Soft links (symbolic links) are references to filenames, while hard links are direct references to the data on disk. Deleting the original file won't affect the soft link, but it will break a hard link.
3. Explain about loops and what are the loops available in LINUX?
   Loops in Linux are used to repeatedly execute a block of code. Common loops are:

1) for loop: Iterates over a sequence.
2) while loop: Executes as long as a condition is true.
3) until loop: Executes until a condition becomes true.

4. What are absolute and relative paths.
Absolute path specifies the full path from the root directory, starting with "/". Relative path specifies the path relative to the current directory.

5. How to debug a shell script.
- Use set -x to enable debug mode.
- Add echo statements to print variables and progress.
- Use set +x to disable debug mode.
- Utilize tools like bash -x script.sh or shellcheck for more advanced debugging.

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Problem Statement a:

Code:

```
: '
Problem Statement :
a) Write a shell program to find Fibonacci series.
'

#!/bin/bash

echo "Enter the number of terms in Fibonacci series:"
read n

a=0
b=1

echo "Fibonacci series:"
for (( i=1; i<=n; i++ ))
do
    echo -n "$a "
    fn=$((a + b))
    a=$b
    b=$fn
done
echo
```

Output:

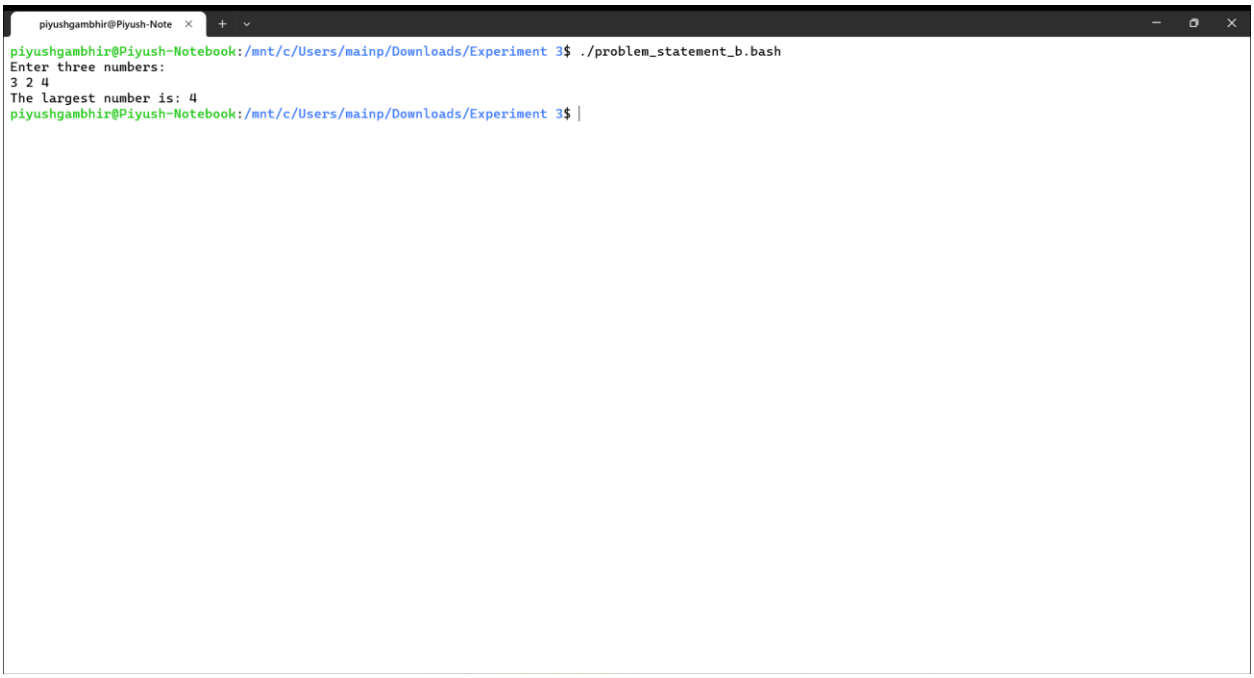

piyushgambhir@Piyush-Note
piyushgambhir@Piyush-Notebook:/mnt/c/Users/mainp/Downloads/Experiment 3$ ./problem_statement_a.bash
Enter the number of terms in Fibonacci series:
5
Fibonacci series:
0 1 1 2 3
piyushgambhir@Piyush-Notebook:/mnt/c/Users/mainp/Downloads/Experiment 3$

Problem Statement b:

Code:

```
: '
Problem Statement :
b) Write a shell program to find the greatest of 3 numbers.
'

#!/bin/bash

echo "Enter three numbers:"
read num1 num2 num3

largest=$num1

if [ $num2 -gt $largest ]; then
    largest=$num2
fi

if [ $num3 -gt $largest ]; then
    largest=$num3
fi

echo "The largest number is: $largest"
```

Output:


piyushgambhir@Piyush-Note
piyushgambhir@Piyush-Notebook:/mnt/c/Users/mainp/Downloads/Experiment 3$ ./problem_statement_b.bash
Enter three numbers:
3 2 4
The largest number is: 4
piyushgambhir@Piyush-Notebook:/mnt/c/Users/mainp/Downloads/Experiment 3$

Problem Statement c:

Code:

```
: '
Problem Statement :
c) Write a shell program to find average of N numbers.
'

#!/bin/bash

echo "Enter the value of N:"
read n

sum=0

echo "Enter $n numbers:"
for (( i=1; i<=n; i++ ))
do
    read num
    sum=$((sum + num))
done

average=$((sum / n))

fractional_part=$((sum % n))
fractional_part=".${fractional_part}"

echo "The average is: $average$fractional_part"
```

Output:


piyushgambhir@Piyush-Note
piyushgambhir@Piyush-Notebook:/mnt/c/Users/mainp/Downloads/Experiment 3$ ./problem_statement_c.bash
Enter the value of N:
5
Enter 5 numbers:
1
2
3
4
5
The average is: 3.0
piyushgambhir@Piyush-Notebook:/mnt/c/Users/mainp/Downloads/Experiment 3$

EXPERIMENT NO. 4

Student Name and Roll Number: Piyush Gambhir
Semester /Section: Semester V – AIML-B (AL-3)
Link to Code: NCU-Lab-Manual-And-End-Semester-Projects/NCU-CSL303-OS-Lab_Manual at main · Piyush-Gambhir/NCU-Lab-Manual-And-End-Semester-Projects (github.com)
Date: 23.08.2023
Faculty Signature:
Marks:

Objective:
To write the shell programming code for the following.

Outcome:
Student is able to write code in shell programming

Problem Statement:
a) Write a shell program to check whether a number is even or odd
b) Write a shell program to find whether a number is prime or not.
c) Write a shell program to find whether a number is palindrome or not.
d) Write a shell program to type number 1 to 7 and then print its corresponding day of week

Background Study:

A shell script is a file with a set of commands in it. The shell reads this file and executes the instructions as if they were input directly on the command line.

A shell is a command-line interpreter and operations such as file manipulation, program execution and text printing is performed by shell script. So, we will use vi editor to edit our files.

Question Bank:

1. What are Zoombie Process?
   A zombie process is a state in which a child process has completed its execution but still has an entry in the process table. This occurs because the process's exit status and resource information are still needed by its parent process to perform cleanup tasks. Zombie processes do not consume system resources but may clutter the process table until the parent process acknowledges their termination through system calls like wait() or waitpid().

2. What are different types of variables used in shell script?
   - **Local Variables:** Used within a function and have a limited scope.
   - **Environment Variables:** Available to all processes in a shell session and can be passed to child processes.

- **Shell Variables:** Internal shell variables that affect its behavior and characteristics.
- **Positional Parameters:** Arguments passed to a script or function when it's called.
- **Special Variables:** Reserved variables like $0 (script name), $# (number of arguments), $? (exit status of the last command), etc.

3. What are the different types of modes available in Vi editor?

- Normal Mode: Default mode for navigation, copying, cutting, and pasting text.
- Insert Mode: Mode for directly entering and editing text.
- Visual Mode: Allows selecting and manipulating text blocks visually.
- Command-Line Mode: Used for saving, quitting, searching, and executing other commands.

4. What are the different types of permission at file level in shell?

- Read (r): Allows reading the contents of a file.
- Write (w): Permits modifying the contents of a file.
- Execute (x): Grants the ability to execute a file (for scripts and binaries).
- Each permission can be assigned separately to the owner, group, and others.

5. How to use comments in shell script.

Use # to begin a comment in a shell script.
Comments are ignored by the shell and are meant for human-readable explanations.
Example: # This is a comment explaining the purpose of the following line.

Student Work Area

Algorithm/Flowchart/Code/Sample output

Problem Statement c:

Code:

```
: '
Problem Statement:
a) Write a shell program to check whether a number is even or odd
'

read -p "Enter a number: " num

if [ $((num%2)) -eq 0 ]
then
    echo "The number is even"
else
    echo "The number is odd"
fi
```

Output:


piyushgambhir@Piyush-Note
piyushgambhir@Piyush-Notebook:/mnt/c/Users/mainp/Downloads/Experiment 4$ ./problem_statement_a.sh
Enter a number: 5
The number is odd
piyushgambhir@Piyush-Notebook:/mnt/c/Users/mainp/Downloads/Experiment 4$

Problem Statement c:

Code:

```
: '
Problem Statement:
b) Write a shell program to find whether a number is prime or not.
'

read -p "Enter a number: " num

if [ $num -eq 1 ]
then
    echo "The number is neither prime nor composite"
    exit
fi

for (( i=2; i<=num/2; i++ ))
do
    if [ $((num%i)) -eq 0 ]
    then
        echo "The number is not prime"
        exit
    fi
done
```

Output:


piyushgambhir@Piyush-Note
piyushgambhir@Piyush-Notebook:/mnt/c/Users/mainp/Downloads/Experiment 4$ ./problem_statement_b.sh
Enter a number: 6
The number is not prime
piyushgambhir@Piyush-Notebook:/mnt/c/Users/mainp/Downloads/Experiment 4$

Problem Statement c:

Code:

```
: '
Problem Statement:
c) Write a shell program to find whether a number is palindrome or not.
'

read -p "Enter a number: " num

temp=$num
rev=0

while [ $num -gt 0 ]
do
    rem=$((num%10))
    rev=$((rev*10+rem))
    num=$((num/10))
done

if [ $temp -eq $rev ]
then
    echo "The number is palindrome"
else
    echo "The number is not palindrome"
fi
```

Output:


piyushgambhir@Piyush-Note
piyushgambhir@Piyush-Notebook:/mnt/c/Users/mainp/Downloads/Experiment 4$ ./problem_statement_c.sh
Enter a number: 56
The number is not palindrome
piyushgambhir@Piyush-Notebook:/mnt/c/Users/mainp/Downloads/Experiment 4$

Problem Statement c:

Code:

```
: '
Problem Statement:
d) Write a shell program to type number 1 to 7 and then print its corresponding
day of week
'

read -p "Enter a number: " num

case $num in
    1)
        echo "Monday"
        ;;
    2)
        echo "Tuesday"
        ;;
    3)
        echo "Wednesday"
        ;;
    4)
        echo "Thursday"
        ;;
    5)
        echo "Friday"
        ;;
    6)
        echo "Saturday"
        ;;
    7)
        echo "Sunday"
        ;;
    *)
        echo "Invalid input"
        ;;
esac
```

Output:


piyushgambhir@Piyush-Note
piyushgambhir@Piyush-Notebook:/mnt/c/Users/mainp/Downloads/Experiment 4$ ./problem_statement_d.sh
Enter a number: 5
Friday
piyushgambhir@Piyush-Notebook:/mnt/c/Users/mainp/Downloads/Experiment 4$

Experiment No: 5

Student Name and Roll Number: Piyush Gambhir
Semester /Section: Semester V – AIML-B (AL-3)
Link to Code: NCU-Lab-Manual-And-End-Semester-Projects/NCU-CSL303-OS-Lab_Manual at main · Piyush-Gambhir/NCU-Lab-Manual-And-End-Semester-Projects (github.com)
Date: 06.09.2023
Faculty Signature:
Marks:

Objective:
Write a program to implement CPU scheduling for first come first serve approach.

Outcome:
The students will understand the First-cum-first-serve algorithm

Problem Statement:
Implement the following CPU scheduling Algorithms.

i) FCFS with Arrival time
ii) FCFS without Arrival time

Background Study:
FCFS

- The simplest CPU-scheduling algorithm is the first-come, first-served (FCFS) scheduling algorithm. With this algorithm, processes are assigned the CPU in the order they request it.
- There is a single queue of ready processes.
- The implementation of the FCFS policy is easily managed with a FIFO queue. When a process enters the ready queue, its PCB is linked onto the tail of the queue.
- The average waiting time under the FCFS policy, however, is often quite long.

Question Bank:

1. Which module gives control of the CPU to the process selected by the short-term scheduler?
   a) **dispatcher**
   b) interrupt
   c) scheduler
   d) none of the mentioned

2. The processes that are residing in main memory and are ready and waiting to execute are kept on a list called
   a) job queue
   b) ready queue
   c) execution queue
   d) process queue

3. The interval from the time of submission of a process to the time of completion is termed as

a) waiting time
b) turnaround time
c) response time
d) throughput

4. Which scheduling algorithm allocates the CPU first to the process that requests the CPU first?
a) first-come, first-served scheduling
b) shortest job scheduling
c) priority scheduling
d) none of the mentioned

5. In priority scheduling algorithm

a) CPU is allocated to the process with highest priority
b) CPU is allocated to the process with lowest priority
c) equal priority processes cannot be scheduled
d) none of the mentioned

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Experiment 5

Problem Statement:

Implement the following CPU scheduling Algorithms.

1. FCFS with Arrival time
2. FCFS without Arrival time

Code:

```python
# importing requried libraries
import pandas as pd
```

[15] Python

```python
def fcfs_with_arrival_time(processes, arrival_time, burst_time):
    n = len(processes)
    wait_time = [0] * n
    turn_around_time = [0] * n
    response_time = [0] * n

    # sorting processes by arrival time
    sorted_indices = sorted(range(n), key=lambda k: arrival_time[k])

    completion_time = 0

    for i in sorted_indices:
        # wait time for a process is completion_time of previous process - arrival_time
        wait_time[i] = completion_time - arrival_time[i]

        # turn around time = wait_time + burst_time
        turn_around_time[i] = wait_time[i] + burst_time[i]

        # update completion_time
        completion_time += burst_time[i]

        # response time =
    df = pd.DataFrame({
        'Processes': processes,
        'Arrival Time': arrival_time,
        'Burst Time': burst_time,
        'Waiting Time': wait_time,
        'Turn-Around Time': turn_around_time
    })
    print(df.to_markdown())
    print("\n")
    print('Average Waiting Time:', sum(wait_time) / n)
    print('Average Turn-Around Time:', sum(turn_around_time) / n)
    print("Completion Time:", completion_time)


processes = [1, 2, 3, 4]
arrival_time = [2, 1, 0, 4]
burst_time = [6, 8, 3, 4]


# waiting time negive (edge case)
# processes = [1, 2, 3]
# arrival_time = [0, 5, 8]
# burst_time = [4, 2, 8]

print('FCFS with Arrival Time')
fcfs_with_arrival_time(processes, arrival_time, burst_time)
```

[21] Python

```
FCFS with Arrival Time
|    |   Processes |   Arrival Time |   Burst Time |   Waiting Time |   Turn-Around Time |
|---:|------------:|---------------:|-------------:|---------------:|-------------------:|
|  0 |           1 |              2 |            6 |              9 |                 15 |
|  1 |           2 |              1 |            8 |              2 |                 10 |
|  2 |           3 |              0 |            3 |              0 |                  3 |
|  3 |           4 |              4 |            4 |             13 |                 17 |


Average Waiting Time: 6.0
Average Turn-Around Time: 11.25
Completion Time: 21
```

```
def fcfs_without_arrival_time(processes, burst_time):
    n = len(processes)
    wait_time = [0] * n
    turn_around_time = [0] * n

    # Calculate waiting time
    for i in range(1, n):
        wait_time[i] = wait_time[i-1] + burst_time[i-1]

    # Calculate turn-around time
    for i in range(n):
        turn_around_time[i] = wait_time[i] + burst_time[i]

    df = pd.DataFrame({
        'Processes': processes,
        'Burst Time': burst_time,
        'Waiting Time': wait_time,
        'Turn-Around Time': turn_around_time
    })
    print(df.to_markdown())
    print("\n")
    print('Average Waiting Time:', sum(wait_time) / n)
    print('Average Turn-Around Time:', sum(turn_around_time) / n)
    print("Completion Time:", sum(burst_time))


# Driver code
processes = [1, 2, 3]
burst_time = [24, 3, 3]

fcfs_without_arrival_time(processes, burst_time)
```

```
|    |   Processes |   Burst Time |   Waiting Time |   Turn-Around Time |
|---:|------------:|-------------:|---------------:|-------------------:|
|  0 |           1 |           24 |              0 |                 24 |
|  1 |           2 |            3 |             24 |                 27 |
|  2 |           3 |            3 |             27 |                 30 |


Average Waiting Time: 17.0
Average Turn-Around Time: 27.0
Completion Time: 30
```

Experiment No: 6

Student Name and Roll Number: Piyush Gambhir
Semester /Section: Semester V – AIML-B (AL-3)
Link to Code: NCU-Lab-Manual-And-End-Semester-Projects/NCU-CSL303-OS-Lab_Manual at main · Piyush-Gambhir/NCU-Lab-Manual-And-End-Semester-Projects (github.com)
Date: 13.09.2023
Faculty Signature:
Marks:

Objective:

Write a program to implement CPU scheduling for shortest job first (Preemptive and Non-Preemptive)

Outcome:
The students will understand the Shortest Job First scheduling mechanism

Problem Statement:
Implement the following CPU scheduling Algorithms.

- SJF (Non-Preemptive)
- SJTF (shortest remaining time first -Preemptive SJF)

Background Study:

- Shortest Job first is having the advantage of a minimum average waiting time .
- This algorithm associates with each process the length of the process next burst time.When CPU is available it assigned to the process that has the smallest next CPU burst time.if CPU burst time of two process is same then it follows FCFS.
- It may cause starvation if shorter processes keep coming. This problem can be solved using the concept of ageing.
- It is practically infeasible as Operating System may not know burst time and therefore may not sort them.

Question Bank:

1. scheduling algorithm In multilevel feedback
   A. processes are not classified into groups
   B. a process can move to a different classified ready queue…
   C. classification of the ready queue is permanent
   D. none of the mentioned
2. Select one which algorithms tend to minimize the process flow time?
   A. First come First served
   B. Earliest Deadline First
   C. Shortest Job First
   D. Longest Job First
3. The process can be classified into many groups in
   A. shortest job scheduling algorithm
   B. multilevel queue scheduling algorithm
   C. round-robin scheduling algorithm
   D. priority scheduling algorithm

4. The turnaround time for short jobs during multiprogramming is usually Shortened and that for long jobs is slightly _____
   A. Shortened
   B. Unchanged
   C. Lengthened
   D. Shortened
5. Time quantum can be said
   A. multilevel queue scheduling algorithm
   B. round-robin scheduling algorithm
   C. shortest job scheduling algorithm
   D. priority scheduling algorithm

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Experiment 6

Problem Statement:

Implement the following CPU scheduling Algorithms.

- SJF (Non-Preemptive)
- SJTF (shortest remaining time first -Preemptive SJF)

Code:

```python
# importing required libraries
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

[45] Python

Non-Preemptive SJF

```python
# function to generate Gantt chart for SJF (Shortest Job First) algorithm
def generate_sjf_gantt_chart_non_preemptive(process, burst_time):
    # Create a DataFrame
    df = pd.DataFrame({'Process': process, 'BurstTime': burst_time})
    df.sort_values('BurstTime', inplace=True)

    # Initialize variables
    current_time = 0
    gantt_chart = []

    # Simulate non-preemptive SJF scheduling
    for index, row in df.iterrows():
        proc = row['Process']
        b_time = row['BurstTime']

        # Update Gantt chart
        gantt_chart.append((proc, current_time, current_time + b_time))
        current_time += b_time

    # Generate the Gantt chart
    fig, ax = plt.subplots()
    fig.set_size_inches(15, 1)
    for process, start, end in gantt_chart:
        ax.broken_barh([(start, end - start)], (0, 1),
                       facecolors="orange", edgecolor="black")
        ax.text((start + end) / 2, 0.5, process,
                ha='center', va='center', color="black")

    ax.set_yticks([])
    ax.set_xticks(np.arange(0, current_time + 1, 1))
    ax.set_xlabel("Time")
    plt.title("Non-Preemptive SJF Gantt Chart")
    plt.show()
```

[46] Python

```python
# function to computer waiting time for SJF (Shortest Job First) algorithm
def compute_sjf_waiting_time_non_preemptive(process, burst_time):
    # create dataframe
    df = pd.DataFrame({'Process': process, 'BurstTime': burst_time, })

    # sorting the dataframe by BurstTime
    df_sorted = df.sort_values(by='BurstTime')

    df_sorted = df_sorted.reset_index(drop=True)
    df_sorted['WaitingTime'] = 0

    # computing waiting time
    for i in range(1, len(df_sorted)):
        df_sorted.loc[i, 'WaitingTime'] = df_sorted.loc[i - 1, 'BurstTime'] + \
            df_sorted.loc[i - 1, 'WaitingTime']

    # returning waiting time (dictionary of process and waiting time)
    return dict(zip(df_sorted['Process'], df_sorted['WaitingTime']))
```

[47] Python

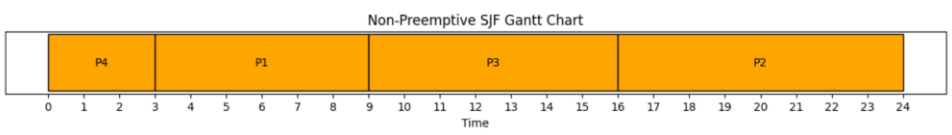
```python
# function to computer turnaround time for non-preemptive SJF (Shortest Job First) algorithm
def compute_sjf_turnaround_time_non_preemptive(process, burst_time):
    n = len(process)
    waiting_time = [0] * n
    turnaround_time = [0] * n

    # calculate waiting time
    waiting_time[0] = 0
    for i in range(1, n):
        waiting_time[i] = burst_time[i-1] + waiting_time[i-1]

    # calculate turnaround time
    for i in range(n):
        turnaround_time[i] = burst_time[i] + waiting_time[i]

    # returning turnaround time (dictionary of process and turnaround time)
    return dict(zip(process, turnaround_time))
```

[48] Python

```python
# defining process and burst time
process = ['P1', 'P2', 'P3', 'P4']
burst_time = [6, 8, 7, 3]

# creating dataframe
df = pd.DataFrame({'Process': process, 'BurstTime': burst_time})

# printing the datafram
print(df.to_markdown(), end="\n\n")

# sorting the dataframe
df = df.sort_values(by='BurstTime')
print(df.to_markdown(), end="\n\n")

# creating the Gantt chart
print("Gantt Chart")
generate_sjf_gantt_chart_non_preemptive(process, burst_time)

# computing waiting time
waiting_time = compute_sjf_waiting_time_non_preemptive(
    process, burst_time)

# computing turnaround time
turn_around_time = compute_sjf_turnaround_time_non_preemptive(
    process, burst_time)

df = pd.DataFrame({'Process': process, 'BurstTime': burst_time,
                   'WaitingTime': list(waiting_time.values()),
                   'TurnAroundTime': list(turn_around_time.values())})


# printing the dataframe
print(df.to_markdown(), end="\n\n")
print(df.to_markdown(), end="\n\n")

# computing average waiting time
print("Average Waiting Time: ", end="")
print(df['WaitingTime'].mean(), end=" milliseconds\n\n")

# computing average turnaround time
print("Average Turnaround Time: ", end="")
print(df['TurnAroundTime'].mean(), end=" milliseconds\n\n")
```

[49] Python

```
|    | Process   |   BurstTime |
|---:|:----------|------------:|
|  0 | P1        |           6 |
|  1 | P2        |           8 |
|  2 | P3        |           7 |
|  3 | P4        |           3 |

|    | Process   |   BurstTime |
|---:|:----------|------------:|
|  3 | P4        |           3 |
|  0 | P1        |           6 |
|  2 | P3        |           7 |
|  1 | P2        |           8 |

Gantt Chart
```




```
|    | Process   |   BurstTime |   WaitingTime |   TurnAroundTime |
|---:|:----------|------------:|--------------:|-----------------:|
|  0 | P1        |           6 |             0 |                6 |
|  1 | P2        |           8 |             3 |               14 |
|  2 | P3        |           7 |             9 |               21 |
|  3 | P4        |           3 |            16 |               24 |

Average Waiting Time: 7.0 milliseconds

Average Turnaround Time: 16.25 milliseconds
```

Preemptive SJF

```python
# function to generate Gantt chart for Preemptive SJF
def generate_sjf_gantt_chart_preemptive(process, arrival_time, burst_time):
    # create dataframe
    df = pd.DataFrame(
        {'Process': process, 'ArrivalTime': arrival_time, 'BurstTime': burst_time})
    df.sort_values(['ArrivalTime', 'BurstTime'], inplace=True)

    # initialize variables
    current_time = 0
    queue = []
    gantt_chart = []
    last_process = None

    # simulate SJF preemptive scheduling
    while not df.empty or queue:
        # add processes that have arrived to the queue
        arrived_processes = df[df['ArrivalTime'] <= current_time]
        queue.extend(arrived_processes.to_dict(orient='records'))
        df.drop(arrived_processes.index, inplace=True)

        # sort the queue based on BurstTime (Shortest Job First)
        queue.sort(key=lambda x: x['BurstTime'])

        if queue:
            # execute the process at the front of the queue
            current_process = queue.pop(0)

            # update Gantt chart
            if last_process == current_process['Process']:
                gantt_chart[-1] = (current_process['Process'],
                                   gantt_chart[-1][1], current_time + 1)
            else:
                gantt_chart.append(
                    (current_process['Process'], current_time, current_time + 1))

            # update current time and burst time for the running process
            current_time += 1
            current_process['BurstTime'] -= 1

            # if the process is finished, update the last process and end time
            last_process = current_process['Process']

            # if the process is not finished, add it back to the queue
            if current_process['BurstTime'] > 0:
                queue.append(current_process)

    # generate the Gantt chart
    fig, ax = plt.subplots()
    fig.set_size_inches(15, 1)
    for process, start, end in gantt_chart:
        ax.broken_barh([(start, end - start)], (0, 1),
                       facecolors=f"orange", edgecolor="black")
        ax.text((start + end) / 2, 0.5, process,
                ha='center', va='center', color="black")

    ax.set_yticks([])
    ax.set_xticks(np.arange(0, current_time + 1, 1))
    ax.set_xlabel("Time")
    plt.title("Preemptive SJF Gantt Chart")
    plt.show()
```

[50] Python

```python
# function to compute waiting time for preemptive SJF
def compute_sjf_waiting_time_preemptive(process, arrival_time, burst_time):
    # create dataframe
    df = pd.DataFrame(
        {'Process': process, 'ArrivalTime': arrival_time, 'BurstTime': burst_time})
    df.sort_values(['ArrivalTime', 'BurstTime'], inplace=True)

    # initialize variables
    current_time = 0
    queue = []
    waiting_time = {p: 0 for p in process}

    # simulating SJF preemptive scheduling
    while not df.empty or queue:
        # add processes that have arrived to the queue
        arrived_processes = df[df['ArrivalTime'] <= current_time]
        queue.extend(arrived_processes.to_dict(orient='records'))
        df.drop(arrived_processes.index, inplace=True)

        # sort the queue based on BurstTime
        queue.sort(key=lambda x: x['BurstTime'])

        if queue:
            # execute the process at the front of the queue
            current_process = queue.pop(0)

            # update waiting time for all processes in the queue
            for q_process in queue:
                waiting_time[q_process['Process']] += 1

            # update current time and burst time for the running process
            current_time += 1
            current_process['BurstTime'] -= 1

            # if the process is not finished, add it back to the queue
            if current_process['BurstTime'] > 0:
                queue.append(current_process)

    # return waiting time (dictionary of process and waiting time)
    return waiting_time
```

[51] Python

```python
def compute_sjf_turnaround_time_preemptive(process, arrival_time, burst_time):
    df = pd.DataFrame({'Process': process, 'ArrivalTime': arrival_time,
                       'BurstTime': burst_time, 'RemainingTime': burst_time})
    df.sort_values(['ArrivalTime', 'BurstTime'], inplace=True)

    current_time = 0
    queue = []
    turnaround_time = {p: 0 for p in process}

    while not df.empty or queue:
        # add processes that have arrived to the queue
        arrived_processes = df[df['ArrivalTime'] <= current_time]
        queue.extend(arrived_processes.to_dict(orient='records'))
        df.drop(arrived_processes.index, inplace=True)

        # sort the queue based on remaining time (Shortest Job First)
        queue.sort(key=lambda x: x['RemainingTime'])

        if queue:
            # execute the process at the front of the queue
            current_process = queue.pop(0)

            # update turnaround time for all processes in the queue
            for q_process in queue:
                turnaround_time[q_process['Process']] += 1

            # update current time and remaining time for the running process
            current_time += 1
            current_process['RemainingTime'] -= 1

            # if the process is not finished, add it back to the queue
            if current_process['RemainingTime'] > 0:
                queue.append(current_process)

            # update turnaround time for the current process
            turnaround_time[current_process['Process']] += 1

    # calculate turnaround time for each process
    for p in process:
        turnaround_time[p] += arrival_time[process.index(p)]

    # return turnaround time (dictionary of process and turnaround time)
    return dict(zip(process, turnaround_time.values()))
```

[52] Python

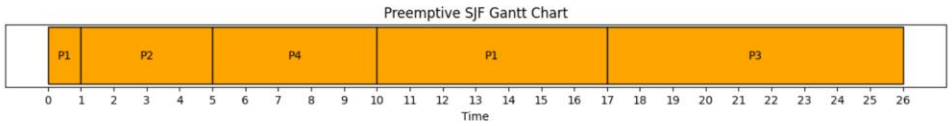
```python
# defining process, arrival time and burst time
process = ['P1', 'P2', 'P3', 'P4']
arrival_time = [0, 1, 2, 3]
burst_time = [8, 4, 9, 5]

# creating dataframe
df = pd.DataFrame(
    {'Process': process, "ArrivalTime": arrival_time, 'BurstTime': burst_time})

# printing the datafram
print(df.to_markdown(), end="\n\n")

# printing the Gannt chart
print("Gantt Chart")
generate_sjf_gantt_chart_preemptive(process, arrival_time, burst_time)

# computing waiting time
waiting_time = compute_sjf_waiting_time_preemptive(
    process, arrival_time, burst_time)

# computing turnaround time
turn_around_time = compute_sjf_turnaround_time_preemptive(
    process, arrival_time, burst_time)

# creating dataframe
df = pd.DataFrame({'Process': process, 'ArrivalTime': arrival_time, 'BurstTime': burst_time,
                   'WaitingTime': list(waiting_time.values()),
                   'TurnAroundTime': list(turn_around_time.values())})

# printing the dataframe
print(df.to_markdown(), end="\n\n")

# computing average waiting time
```

```
print("Average Waiting Time: ", end="")
print(df['WaitingTime'].mean(), end=" milliseconds\n\n")

# computing average turnaround time
print("Average Turnaround Time: ", end="")
print(df['TurnAroundTime'].mean(), end=" milliseconds\n\n")
```

[53] Python

	Process	ArrivalTime	BurstTime
0	P1	0	8
1	P2	1	4
2	P3	2	9
3	P4	3	5

Gantt Chart



	Process	ArrivalTime	BurstTime	WaitingTime	TurnAroundTime
0	P1	0	8	9	17
1	P2	1	4	0	5
2	P3	2	9	15	26
3	P4	3	5	2	10

Average Waiting Time: 6.5 milliseconds

Average Turnaround Time: 14.5 milliseconds

Experiment No: 7

Student Name and Roll Number: Piyush Gambhir
Semester /Section: Semester V – AIML-B (AL-3)
Link to Code: NCU-Lab-Manual-And-End-Semester-Projects/NCU-CSL303-OS-Lab_Manual at main · Piyush-Gambhir/NCU-Lab-Manual-And-End-Semester-Projects (github.com)
Date: 19.09.2023
Faculty Signature:
Marks:

Objective:
Write a program to perform priority scheduling among a set of processes.

Outcome:
Student will understand the working of priority scheduling among a set of processes.

Problem Statement:
Implement the priority scheduling.

Background Study:
Priority scheduling is a non-preemptive algorithm and used in batch systems. Each process is assigned a priority. Process with highest priority is to be executed first and so on.

Question Bank:

1. What are advantages of Priority scheduling?

   Easy to use scheduling method. Processes are executed on the basis of priority so high priority does not need to wait for long which saves time.

2. What are disadvantages of priority scheduling?

   The major disadvantage of priority scheduling is the process of indefinite blocking or starvation. This problem appears when a process is ready to be executed but it has to wait for the long time for execution by CPU because other high priority processes are executed by the CPU.

3. At the ready queue when a process arrives In priority scheduling algorithm, the priority of this process is compared with the priority of?
   A. currently running process
   B. parent process
   C. all process
   D. init process

4. Differentiate between pre-emptive and non pre-emptive scheduling?

   The basic difference between preemptive and non-preemptive scheduling is that in preemptive scheduling the CPU is allocated to the processes for the limited time. While in Non-preemptive scheduling, the CPU is allocated to the process till it terminates or switches to waiting state.

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Experiment 7

Problem Statement:

Implement the priority scheduling.

Code:

```python
# importing required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from queue import PriorityQueue
```

[1] Python

Non-Premeptive Priority Scheduling

```python
# function to generate gantt chart for non-premptive priority scheduling
def generate_priority_gantt_chart_non_preemptive(process, priority):
    df = pd.DataFrame({'Process': process, 'Priority': priority})
    df.sort_values('Priority', inplace=True)
    current_time = 0
    gantt_chart = []

    for index, row in df.iterrows():
        proc = row['Process']
        prio = row['Priority']

        gantt_chart.append((proc, current_time, current_time + prio))
        current_time += prio

    fig, ax = plt.subplots()
    fig.set_size_inches(15, 1)
    for process, start, end in gantt_chart:
        ax.broken_barh([(start, end - start)], (0, 1),
                       facecolors="orange", edgecolor="black")
        ax.text((start + end) / 2, 0.5, process,
                ha='center', va='center', color="black")

    ax.set_yticks([])
    ax.set_xticks(np.arange(0, current_time + 1, 1))
    ax.set_xlabel("Time")
    plt.title("Non-Preemptive Priority Gantt Chart")
    plt.show()
```

[2] Python

```python
# function to compute waiting time of all processes for non-preemptive priority scheduling
def compute_priority_waiting_time_non_preemptive(process, burst_time, priority):
    df = pd.DataFrame(
        {'Process': process, 'BurstTime': burst_time, 'Priority': priority})
    df_sorted = df.sort_values(by='Priority')
    df_sorted = df_sorted.reset_index(drop=True)
    df_sorted['WaitingTime'] = 0

    for i in range(1, len(df_sorted)):
        df_sorted.loc[i, 'WaitingTime'] = df_sorted.loc[i - 1,
                                                        'BurstTime'] + df_sorted.loc[i - 1, 'WaitingTime']

    return dict(zip(df_sorted['Process'], df_sorted['WaitingTime']))
```

[3] Python

```python
# function to turn around time of all processes for non-preemptive priority scheduling
def compute_priority_turnaround_time_non_preemptive(process, burst_time, priority):
    n = len(process)
    df = pd.DataFrame(
        {'Process': process, 'BurstTime': burst_time, 'Priority': priority})
    df_sorted = df.sort_values(by='Priority')
    df_sorted = df_sorted.reset_index(drop=True)

    waiting_time = [0] * n
    turnaround_time = [0] * n

    waiting_time[0] = 0
    for i in range(1, n):
        waiting_time[i] = df_sorted.loc[i - 1, 'BurstTime'] + waiting_time[i-1]

    for i in range(n):
        turnaround_time[i] = df_sorted.loc[i, 'BurstTime'] + waiting_time[i]
```

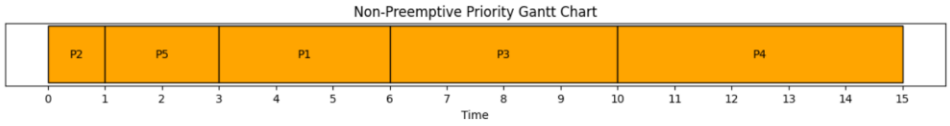
```
    return dict(zip(df_sorted['Process'], turnaround_time))
```

[4] Python

```
# defining the processes, burst time and priority
process = ['P1', 'P2', 'P3', 'P4', 'P5']
burst_time = [10, 1, 2, 1, 5]
priority = [3, 1, 4, 5, 2]


# generating gantt chart for priority scheduling
generate_priority_gantt_chart_non_preemptive(process, priority)


# computing waiting time of all processes
waiting_time = compute_priority_waiting_time_non_preemptive(
    process, burst_time, priority)

# computing turnaround time of all processes
turn_around_time = compute_priority_turnaround_time_non_preemptive(
    process, burst_time, priority)

# generating dataframe for with waiting time and turnaround time
df = pd.DataFrame({'Process': process, 'BurstTime': burst_time, 'Priority': priority,
                   'WaitingTime': list(waiting_time.values()),
                   'TurnAroundTime': list(turn_around_time.values())})
print(df.to_markdown(), end="\n\n")


# computing and printing average waiting time
print("Average Waiting Time: ", df['WaitingTime'].mean(), "milliseconds\n")
```

```
# computing and printing average turnaround time
print("Average Turnaround Time: ",
      df['TurnAroundTime'].mean(), "milliseconds\n")
```

[5] Python



	Process	BurstTime	Priority	WaitingTime	TurnAroundTime
0	P1	10	3	0	1
1	P2	1	1	1	6
2	P3	2	4	6	16
3	P4	1	5	16	18
4	P5	5	2	18	19

Average Waiting Time: 8.2 milliseconds

Average Turnaround Time: 12.0 milliseconds

Premeptive Priority Scheduling

```
class Process:
    def __init__(self, name, arrival_time, burst_time, priority):
        self.name = name
        self.arrival_time = arrival_time
        self.burst_time = burst_time
        self.priority = priority
        self.remaining_time = burst_time

def non_preemptive_priority_scheduling(processes):
    processes.sort(key=lambda x: (x.priority, x.arrival_time))
    n = len(processes)
    completion_time = [0] * n
    turnaround_time = [0] * n
    waiting_time = [0] * n

    for i in range(n):
        if i == 0:
            completion_time[i] = processes[i].arrival_time + processes[i].burst_time
        else:
            completion_time[i] = max(processes[i].arrival_time, completion_time[i - 1]) + processes[i].burst_time

        turnaround_time[i] = completion_time[i] - processes[i].arrival_time
        waiting_time[i] = turnaround_time[i] - processes[i].burst_time

    total_waiting_time = sum(waiting_time)
    avg_waiting_time = total_waiting_time / n

    print("Non-preemptive Priority Scheduling:")
    print("Process\tArrival Time\tBurst Time\tPriority\tCompletion Time\tTurnaround Time\tWaiting Time")
    for i in range(n):
```

```
        p = processes[i]
        print(f"{p.name}\t{p.arrival_time}\t\t{p.burst_time}\t\t{p.priority}\t\t{completion_time[i]}\t\t{turnaround_time[i]}\t\t{waiting_time[i]}")
    print(f"Average Waiting Time: {avg_waiting_time:.2f}")

def preemptive_priority_scheduling(processes):
    n = len(processes)
    current_time = 0
    completion_time = [0] * n
    turnaround_time = [0] * n
    waiting_time = [0] * n

    while True:
        remaining_processes = [p for p in processes if p.remaining_time > 0 and p.arrival_time <= current_time]

        if not remaining_processes:
            break

        highest_priority_process = min(remaining_processes, key=lambda x: x.priority)
        index = processes.index(highest_priority_process)

        processes[index].remaining_time -= 1
        current_time += 1

        if processes[index].remaining_time == 0:
            completion_time[index] = current_time
            turnaround_time[index] = completion_time[index] - processes[index].arrival_time
            waiting_time[index] = turnaround_time[index] - processes[index].burst_time

    total_waiting_time = sum(waiting_time)
    avg_waiting_time = total_waiting_time / n

    print("\nPreemptive Priority Scheduling:")
    print("Process\tArrival Time\tBurst Time\tPriority\tCompletion Time\tTurnaround Time\tWaiting Time")
```

```
    for i in range(n):
        p = processes[i]
        print(f"{p.name}\t{p.arrival_time}\t\t{p.burst_time}\t\t{p.priority}\t\t{completion_time[i]}\t\t{turnaround_time[i]}\t\t{waiting_time[i]}")
    print(f"Average Waiting Time: {avg_waiting_time:.2f}")

if __name__ == "__main__":
    processes = [
        Process("P1", 0, 4, 2),
        Process("P2", 1, 3, 1),
        Process("P3", 2, 1, 3),
        Process("P4", 3, 2, 4)
    ]

    non_preemptive_priority_scheduling(processes)
    preemptive_priority_scheduling(processes)
```

[6] Python

```
Non-preemptive Priority Scheduling:
Process Arrival Time    Burst Time      Priority        Completion Time Turnaround Time Waiting Time
P2      1               3               1               4               3               0
P1      0               4               2               8               8               4
P3      2               1               3               9               7               6
P4      3               2               4               11              8               6
Average Waiting Time: 4.00

Preemptive Priority Scheduling:
Process Arrival Time    Burst Time      Priority        Completion Time Turnaround Time Waiting Time
P2      1               3               1               4               3               0
P1      0               4               2               7               7               3
P3      2               1               3               8               6               5
P4      3               2               4               10              7               5
Average Waiting Time: 3.25
```

Experiment No: 8

Student Name and Roll Number: Piyush Gambhir
Semester /Section: Semester V – AIML-B (AL-3)
Link to Code: NCU-Lab-Manual-And-End-Semester-Projects/NCU-CSL303-OS-Lab_Manual at main · Piyush-Gambhir/NCU-Lab-Manual-And-End-Semester-Projects (github.com)
Date: 27.09.2023
Faculty Signature:
Marks:

Objective: Objective

To familiarize the students about CPU scheduling Algorithms

Program Outcome

The students will understand the Round Robin Algorithm.

Problem Statement:
Implement the Round Robin Algorithm.

Background Study:

- In Round Robin each process is assigned a fixed time slot in a cyclic way and this is preemptive. It has a disadvantage of context switch and have quantum time

Question Bank:

1. What is Preemptive and Non- Preemptive CPU scheduling? Explain with examples.

   Preemptive:

   In this resources(CPU Cycle) are allocated to a process for a limited time.
   Eg. Examples of preemptive scheduling are Round Robin and Shortest Remaining Time First.

   Non-Preemptive

   Once resources(CPU Cycle) are allocated to a process, the process holds it till it completes its burst time or switches to waiting state.
   Eg. Examples of non-preemptive scheduling are First Come First Serve and Shortest Job First.
2. Explain the difference between short term, long term and medium term scheduling.
   - Long term: It selects processes from the pool and load them into memory for execution.
   - Medium term: Process can be reintroduced into the meat and its execution can be continued.
   - Short term: It selects from among the processes that are ready to execute.

3. Explain the function of Dispatcher and Context Switch mechanism.

   The dispatcher in an operating system plays a crucial role in managing the transition of the CPU from one process to another. It ensures that processes are executed in a fair and efficient manner by performing context switches, which involve saving the state of the currently running process and loading the state of a new process. This seamless handover of control allows the operating system to efficiently multitask and provide a responsive environment for users and applications..

4. What are the advantages and disadvantages of Round robin?

   Advantages

   - Every process gets an equal share of the CPU.
   - RR is cyclic in nature, so there is no starvation.

   Disadvantages

   - Setting the quantum too short, increases the overhead and lowers the CPU efficiency, but setting it too long may cause poor response to short processes.
   - Average waiting time under the RR policy is often long.

5. Give the application are of Robin Robin.

   A popular use of the queue data structure is the scheduling problem in the operating system. Round-robin is one of the simplest scheduling algorithms for processes in an operating system, which assigns time slices to each process in equal portions and in order, handling all processes without priority.

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Experiment 8

Problem Statement:

Implement the Round Robin Algorithm.

Code:

```python
# importing required libraries
import pandas as pd
import matplotlib.pyplot as plt
```

[1] Python

```python
# function to generate the gantt chart for round robin scheduling
def generate_rr_gantt_chart(processes, burst_time, time_quantum):
    # Create DataFrame
    df = pd.DataFrame({'Process': processes, 'BurstTime': burst_time})
    n = len(processes)

    # Initialize Gantt chart
    fig, ax = plt.subplots()
    fig.set_size_inches(15, 1)

    # Variables for time and labels
    current_time = 0
    labels = []
    rem_bt = [0] * n

    for i in range(n):
        rem_bt[i] = burst_time[i]

    # Create Gantt chart
    while True:
        done = True
        for i, row in df.iterrows():
            if rem_bt[i] > 0:
                done = False
                bt = min(time_quantum, rem_bt[i])
                ax.broken_barh([(current_time, bt)], (0, 1),
                               facecolors='orange', edgecolor='black')
                ax.text(current_time + bt / 2 - 0.1, 0.5,
                        row['Process'], fontsize=12, verticalalignment='center')
                labels.append(str(current_time))
                current_time += bt
                rem_bt[i] -= bt

        if done:
            break

    labels.append(str(current_time))

    # Chart attributes
    ax.set_yticks([])
    ax.set_xticks([int(label) for label in labels])
    ax.set_xticklabels(labels)
    ax.set_xlabel("Time")

    plt.title("Round Robin Gantt Chart")
    plt.show()
```

```python
# function to compute the waiting time for round robin
def waiting_time_rr(processes, burst_time, time_quantum):
    n = len(processes)
    waiting_time = [0] * n
    remaining_burst_time = [0] * n
    for i in range(n):
        remaining_burst_time[i] = burst_time[i]

    t = 0
    while True:
        done = True
        for i in range(n):
            if remaining_burst_time[i] > 0:
                done = False
                if remaining_burst_time[i] > time_quantum:
                    t += time_quantum
                    remaining_burst_time[i] -= time_quantum
                else:
                    t += remaining_burst_time[i]
                    waiting_time[i] = t - burst_time[i]
                    remaining_burst_time[i] = 0
        if done:
            break

    # returning the waiting time (dictionary with process as key and waiting time)
    return {processes[i]: waiting_time[i] for i in range(n)}
```

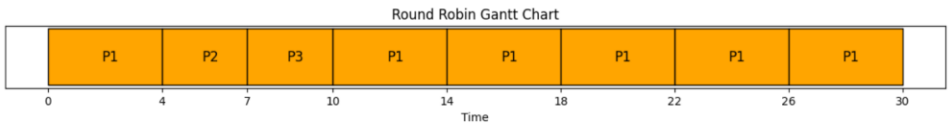
[3] Python

```python
# function to compute the turnaround time for round robin
def turnaround_time_rr(processes, burst_time, waiting_time):
    n = len(processes)
    turnaround_time = [0] * n
    turnaround_time_dict = {}

    for i in range(n):
        turnaround_time[i] = burst_time[i] + waiting_time[processes[i]]
        turnaround_time_dict[processes[i]] = turnaround_time[i]

    # return the turnaround time (dictionary of processes and their turnaround time)
    return turnaround_time_dict
```

[4] Python

```python
# defining processes and burst time
processes = ['P1', 'P2', 'P3']
burst_time = [24, 3, 3]

# defining time quantum
time_quantum = 4

# generating the gantt chart
generate_rr_gantt_chart(processes, burst_time, time_quantum)

# computing waiting time
waiting_time = waiting_time_rr(processes, burst_time, time_quantum)

# computing turnaround time
turnaround_time = turnaround_time_rr(processes, burst_time, waiting_time)

# creting a dataframe of processes, burst time, waiting time and turnaround time
df = pd.DataFrame({'Process': processes, 'Burst Time': burst_time, 'Waiting Time': list(
    waiting_time.values()), 'Turnaround Time': list(turnaround_time.values())})

# printing the dataframe
print(df.to_markdown(index=False))

# computing the average waiting time
avg_waiting_time = sum(list(waiting_time.values())) / len(waiting_time)
print(f'Average Waiting Time: {avg_waiting_time}')

# computing the average turnaround time
avg_turnaround_time = sum(
    list(turnaround_time.values())) / len(turnaround_time)
print(f'Average Turnaround Time: {avg_turnaround_time}')
```

[5] Python



Process	Burst Time	Waiting Time	Turnaround Time
P1	24	6	30
P2	3	4	7
P3	3	7	10

Average Waiting Time: 5.666666666666667
Average Turnaround Time: 15.666666666666666

Experiment No: 9

Student Name and Roll Number: Piyush Gambhir
Semester /Section: Semester V – AIML-B (AL-3)
Link to Code: NCU-Lab-Manual-And-End-Semester-Projects/NCU-CSL303-OS-Lab_Manual at main · Piyush-Gambhir/NCU-Lab-Manual-And-End-Semester-Projects (github.com)
Date: 04.10.2023
Faculty Signature:
Marks:

Objective: Write a program to implement reader/writer problem using semaphore

Program Outcome

The students will understand the reader/writer problem using semaphore

Problem Statement:
Write a program to implement reader/writer problem using semaphore.

Background Study: There is a shared resource which should be accessed by multiple processes. There are two types of processes in this context. They are reader and writer. Any number of readers can read from the shared resource simultaneously, but only one writer can write to the shared resource. When a writer is writing data to the resource, no other process can access the resource. A writer cannot write to the resource if there are non-zero number of readers accessing the resource at that time.

Question Bank:

1. An un-interruptible unit is known as _____
   a) single
   b) atomic
   c) static
   d) none of the mentioned
2. TestAndSet instruction is executed _____
   a) after a particular process
   b) periodically
   c) atomically
   d) none of the mentioned
3. Semaphore is a/an _____ to solve the critical section problem.
   a) hardware for a system
   b) special program for a system
   c) integer variable
   d) none of the mentioned
4. What are the two atomic operations permissible on semaphores?
   a) wait
   b) stop

c) hold
d) none of the mentioned

5. When several processes access the same data concurrently and the outcome of the execution depends on the particular order in which the access takes place is called _____
a) dynamic condition
b) race condition
c) essential condition
d) critical condition

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Experiment 9

Problem Statement:

Write a program to implement reader/writer problem using semaphore.

Code:

```python
# importing required libraries
import threading
import time
```

[8] Python

```python
class ReaderWriterLock:
    def __init__(self):
        self.reader_count = 0
        self.writer_count = 0
        self.reader_lock = threading.Lock()
        self.writer_lock = threading.Lock()
        self.write_request = threading.Lock()
        self.reader_semaphore = threading.Semaphore(3)

    def acquire_read(self):
        self.reader_semaphore.acquire()
        self.write_request.acquire()
        self.reader_lock.acquire()
        self.reader_count += 1
        if self.reader_count == 1:
            self.writer_lock.acquire()
        self.reader_lock.release()
        self.write_request.release()

    def release_read(self):
        self.reader_lock.acquire()
        self.reader_count -= 1
        if self.reader_count == 0:
            self.writer_lock.release()
        self.reader_lock.release()
        self.reader_semaphore.release()

    def acquire_write(self):
        self.write_request.acquire()
        self.writer_lock.acquire()
        self.write_request.release()

    def release_write(self):
        self.writer_lock.release()


stop_threads = False

if __name__ == "__main__":
    rw_lock = ReaderWriterLock()

    def reader(id):
        while not stop_threads:
            rw_lock.acquire_read()
            print(f"Reader-{id} is reading", end="\n")
            time.sleep(1)
            rw_lock.release_read()

    def writer(id):
        while not stop_threads:
            rw_lock.acquire_write()
            print(f"Writer-{id} is writing", end="\n")
            time.sleep(2)
            rw_lock.release_write()

    for i in range(5):
        threading.Thread(target=reader, args=(i,)).start()

    writer_thread = threading.Thread(target=writer, args=(1,))
    writer_thread.start()

    time.sleep(10)
    stop_threads = True
```

[9] Python

```
Reader-0 is reading
Reader-1 is reading
Reader-2 is reading
Writer-1 is writing
Reader-0 is readingReader-1 is reading

Reader-2 is reading
Writer-1 is writing
Reader-1 is readingReader-0 is reading
Reader-2 is reading

Writer-1 is writing
Reader-0 is readingReader-2 is reading

Reader-1 is reading
Writer-1 is writing
Reader-4 is reading
Reader-3 is reading
```

Experiment No: 10

Student Name and Roll Number: Piyush Gambhir
Semester /Section: Semester V – AIML-B (AL-3)
Link to Code: NCU-Lab-Manual-And-End-Semester-Projects/NCU-CSL303-OS-Lab_Manual at main · Piyush-Gambhir/NCU-Lab-Manual-And-End-Semester-Projects (github.com)
Date: 25.10.2023
Faculty Signature:
Marks:

Objective:

Write a program to implement Dining Philosopher's problem using semaphore

Outcome:
The students will understand the problem of synchronization among processes and its solution through Dining Philosopher's problem using semaphore

Problem Statement:

Write a program to implement Dining Philosopher's problem using semaphore.

Background Study:
Five philosophers, spend their time thinking and eating spaghetti. They eat at a round table with five individual seats. For eating each philosopher needs two forks (the resources). There are five forks on the table, one left and one right of each seat. When a philosopher cannot grab both forks it sits and waits. Eating takes random time, then the philosopher puts the forks down and leaves the dining room. After spending some random time thinking he again becomes hungry, and the circle repeats itself.

Question Bank:

1. Which one of the following is a synchronization tool?
   a) thread
   b) pipe
   c) semaphore
   d) socket
2. A semaphore is a shared integer variable _____
   a) that can not drop below zero
   b) that can not be more than zero
   c) that can not drop below one
   d) that can not be more than one
3. The bounded buffer problem is also known as _____
   a) Readers – Writers problem
   b) Dining – Philosophers problem
   c) Producer – Consumer problem
   d) None of the mentioned

4. In the bounded buffer problem _____
   a) there is only one buffer
   b) there are n buffers (n being greater than one but finite)
   c) there are infinite buffers
   d) the buffer size is bounded
5. To ensure difficulties do not arise in the readers – writers problem _____ are given exclusive access to the shared object.
   a) readers
   b) writers
   c) readers and writers
   d) none of the mentioned

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Experiment 10

Problem Statement:

Write a program to implement Dining Philosopher's problem using semaphore.

Code:

```python
# importing required libraries
import threading
import time
import random
```

[2] Python

```python
N = 5  # number of philosophers

# Semaphore for controlling access to the forks
forks = [threading.Semaphore(1) for _ in range(N)]

# Event to signal threads to stop
stop_event = threading.Event()


def philosopher(i):
    while not stop_event.is_set():  # Check if stop_event is set
        think(i)
        take_forks(i)
        eat(i)
        put_forks(i)


def think(i):
    print(f"Philosopher {i} is thinking")
    time.sleep(random.uniform(0.1, 0.5))


def take_forks(i):
    # Acquire forks in increasing order to avoid deadlock
    left = i
    right = (i + 1) % N

    if left < right:
        forks[left].acquire()
        forks[right].acquire()
    else:
        forks[right].acquire()
        forks[left].acquire()


def eat(i):
    print(f"Philosopher {i} is eating")
    time.sleep(random.uniform(0.1, 0.5))


def put_forks(i):
    left = i
    right = (i + 1) % N

    # Release forks
    forks[left].release()
    forks[right].release()


if __name__ == "__main__":
    # Start the philosophers
    philosophers = [threading.Thread(
        target=philosopher, args=(i,)) for i in range(N)]
    for p in philosophers:
        p.start()

    # Signal threads to stop after 10 seconds
    time.sleep(10)
    stop_event.set()

    # Wait for threads to finish
    for p in philosophers:
        p.join()
```

[3] Python

```
Philosopher 0 is thinking
Philosopher 1 is thinking
Philosopher 2 is thinking
Philosopher 3 is thinking
Philosopher 4 is thinking
Philosopher 0 is eating
Philosopher 3 is eating
Philosopher 0 is thinking
Philosopher 3 is thinking
Philosopher 4 is eating
Philosopher 2 is eating
Philosopher 2 is thinkingPhilosopher 1 is eating

Philosopher 4 is thinking
Philosopher 1 is thinkingPhilosopher 0 is eating

Philosopher 3 is eating
Philosopher 3 is thinking
Philosopher 2 is eating
Philosopher 0 is thinking
Philosopher 4 is eating
Philosopher 2 is thinking
Philosopher 1 is eating
Philosopher 4 is thinkingPhilosopher 3 is eating

Philosopher 1 is thinking
Philosopher 0 is eating
Philosopher 0 is thinking
Philosopher 3 is thinkingPhilosopher 2 is eating
Philosopher 4 is eating
```

Experiment No:11

Student Name and Roll Number: Piyush Gambhir
Semester /Section: Semester V – AIML-B (AL-3)
Link to Code: NCU-Lab-Manual-And-End-Semester-Projects/NCU-CSL303-OS-Lab_Manual at main · Piyush-Gambhir/NCU-Lab-Manual-And-End-Semester-Projects (github.com)
Date: 01.11.2023
Faculty Signature:
Marks:

Objective:
Write a program to implement Banker's algorithm for deadlock avoidance.

Outcome:
The students will understand how system handles deadlock using Banker's algorithm for deadlock avoidance

Problem Statement:
Write a program to implement Banker's algorithm for deadlock avoidance.

Background Study:
The banker's algorithm is a resource allocation and deadlock avoidance algorithm that tests for safety by simulating the allocation for predetermined maximum possible amounts of all resources, then makes an "s-state" check to test for possible activities, before deciding whether allocation should be allowed to continue. Banker's algorithm is named so because it is used in banking system to check whether loan can be sanctioned to a person or not. Suppose there are n number of account holders in a bank and the total sum of their money is S. If a person applies for a loan then the bank first subtracts the loan amount from the total money that bank has and if the remaining amount is greater than S then only the loan is sanctioned. It is done because if all the account holders comes to withdraw their money then the bank can easily do it.

Question Bank:

1. Each request requires that the system consider the _____ to decide whether the current request can be satisfied or must wait to avoid a future possible deadlock.
   a) resources currently available
   b) processes that have previously been in the system
   c) resources currently allocated to each process
   d) future requests and releases of each process
2. Given a priori information about the _____ number of resources of each type that maybe requested for each process, it is possible to construct an algorithm that ensures that the system will never enter a deadlock state.
   a) minimum
   b) average
   c) maximum
   d) approximate
3. A deadlock avoidance algorithm dynamically examines the _____ to ensure that a circular wait condition can never exist.
   a) resource allocation state

b) system storage state

c) operating system

d) resources

4. A state is safe, if _____

a) the system does not crash due to deadlock occurrence

b) the system can allocate resources to each process in some order and still avoid a deadlock

c) the state keeps the system protected and safe

d) all of the mentioned

5. The two ways of aborting processes and eliminating deadlocks are _____

a) Abort all deadlocked processes

b) Abort all processes

c) Abort one process at a time until the deadlock cycle is eliminated

d) All of the mentioned

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Experiment 11

Problem Statement:

Write a program to implement Banker's algorithm for deadlock avoidance.

Code:

```python
# importing required libraries
import numpy as np
```

[2] Python

```python
class BankersAlgorithm:
    def __init__(self, num_processes, num_resources):
        self.available = [0] * num_resources
        self.maximum = np.zeros((num_processes, num_resources), dtype=int)
        self.allocated = np.zeros((num_processes, num_resources), dtype=int)
        self.need = np.zeros((num_processes, num_resources), dtype=int)

    def is_request_safe(self, process_id, request_vector):
        if any(request_vector[i] > self.need[process_id][i] for i in range(len(request_vector))):
            return False
        if any(request_vector[i] > self.available[i] for i in range(len(request_vector))):
            return False
        self.available = [self.available[i] - request_vector[i] for i in range(len(request_vector))]
        self.allocated[process_id] = [self.allocated[process_id][i] + request_vector[i] for i in range(len(request_vector))]
        self.need[process_id] = [self.need[process_id][i] - request_vector[i] for i in range(len(request_vector))]
        if self.check_safety():
            return True
        self.available = [self.available[i] + request_vector[i] for i in range(len(request_vector))]
        self.allocated[process_id] = [self.allocated[process_id][i] - request_vector[i] for i in range(len(request_vector))]
        self.need[process_id] = [self.need[process_id][i] + request_vector[i] for i in range(len(request_vector))]
        return False

    def check_safety(self):
        work = list(self.available)
        finish = [False] * len(self.need)

        while True:
            for i, (need, alloc) in enumerate(zip(self.need, self.allocated)):
                if not finish[i] and all(need[j] <= work[j] for j in range(len(work))):
                    work = [work[j] + alloc[j] for j in range(len(work))]
                    finish[i] = True
                    break
            else:
                break

        return all(finish)

if __name__ == "__main__":
    num_processes = 5
    num_resources = 3

    bankers = BankersAlgorithm(num_processes, num_resources)

    bankers.maximum = np.array([[7, 5, 3], [3, 2, 2], [9, 0, 2], [2, 2, 2], [4, 3, 3]])
    bankers.allocated = np.array([[0, 1, 0], [2, 0, 0], [3, 0, 2], [2, 1, 1], [0, 0, 2]])
    bankers.available = [3, 2, 2]
    bankers.need = bankers.maximum - bankers.allocated

    process_id = 1
    request_vector = [1, 0, 2]

    if bankers.is_request_safe(process_id, request_vector):
        print(f"The request from process {process_id} is safe.")
    else:
        print(f"The request from process {process_id} is not safe.")
```

[3] Python

```
The request from process 1 is not safe.
```

Experiment No: 12

Student Name and Roll Number: Piyush Gambhir
Semester /Section: Semester V – AIML-B (AL-3)
Link to Code: NCU-Lab-Manual-And-End-Semester-Projects/NCU-CSL303-OS-Lab_Manual at main · Piyush-Gambhir/NCU-Lab-Manual-And-End-Semester-Projects (github.com)
Date: 08.11.2023
Faculty Signature:

Objective:
Write a program for page replacement policy using a) LRU b) FIFO c) Optimal.

Outcome:
The students will understand concept of memory management through various page replacement policy using a) LRU b) FIFO

Problem Statement:
Write a program to implement page replacement policy using a) LRU b) FIFO c) Optimal.

Background Study:
In operating systems that use paging for memory management, page replacement algorithm are needed to decide which page needed to be replaced when new page comes in. Whenever a new page is referred and not present in memory, page fault occurs and Operating System replaces one of the existing pages with newly needed page. Different page replacement algorithms suggest different ways to decide which page to replace. The target for all algorithms is to reduce number of page faults.

Question Bank:

1. The main memory accommodates _____
a) operating system
b) cpu
c) user processes
d) all of the mentioned

2. What is the operating system?
a) in the low memory
b) in the high memory
c) either low or high memory (depending on the location of interrupt vector)
d) none of the mentioned

3. In contiguous memory allocation _____
a) each process is contained in a single contiguous section of memory
b) all processes are contained in a single contiguous section of memory
c) the memory space is contiguous
d) none of the mentioned

4. The relocation register helps in _____
a) providing more address space to processes
b) a different address space to processes
c) to protect the address spaces of processes
d) none of the mentioned

5. With relocation and limit registers, each logical address must be _____ the limit register.
a) less than
b) equal to
c) greater than
d) none of the mentioned

Student Work Area

Algorithm/Flowchart/Code/Sample Outputs

Experiment 12

Problem Statement:

Write a program to implement page replacement policy using

1. LRU
2. FIFO
3. Optimal

Code:

```python
# importing required libraries
from collections import deque
```

```python
def lru(pages, n, capacity):
    s = []
    page_faults = 0

    for i in pages:
        if i not in s:
            if len(s) == capacity:
                s.remove(s[0])
            s.append(i)
            page_faults += 1
        else:
            s.remove(i)
            s.append(i)

    return page_faults


def fifo(pages, n, capacity):
    queue = []
    page_faults = 0

    for i in pages:
        if i not in queue:
            if len(queue) == capacity:
                queue.pop(0)
            queue.append(i)
            page_faults += 1

    return page_faults
```

```python
def optimal(pages, n, capacity):
    s = []
    page_faults = 0
    occurance = [None for i in range(capacity)]

    for i in range(n):
        if pages[i] not in s:
            if len(s) < capacity:
                s.append(pages[i])5
            else:
                for x in range(len(s)):
                    if s[x] not in pages[i+1:]:
                        occurance[x] = 999999
                    else:
                        occurance[x] = pages[i+1:].index(s[x])
                s[occurance.index(max(occurance))] = pages[i]
            page_faults += 1

    return page_faults


if __name__ == "__main__":
    pages = [7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0]
    n = len(pages)
    capacity = 3

    print(f"Page Faults using LRU: {lru(pages, n, capacity)}")
    print(f"Page Faults using FIFO: {fifo(pages, n, capacity)}")
    print(f"Page Faults using Optimal: {optimal(pages, n, capacity)}")
```

```
Page Faults using LRU: 12
Page Faults using FIFO: 14
Page Faults using Optimal: 9
```