Exp.9

**Problem Statement:**

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

CODE:-

#!/bin/bash

# File locks for semaphores

read\_count\_file="read\_count.txt"

rw\_mutex\_file="rw\_mutex.lock"

mutex\_file="mutex.lock"

# Initialize semaphores and shared variable

echo 0 > "$read\_count\_file" # Initialize read count to 0

touch "$rw\_mutex\_file" # Semaphore for writer access

touch "$mutex\_file" # Semaphore for read count modification

shared\_data="shared\_data.txt"

echo "Initial data" > "$shared\_data" # Shared resource

# Reader function

reader() {

while true; do

echo "Reader $(($1)) is trying to read..."

# Entry section

exec {mutex\_fd}>"$mutex\_file" # Lock mutex

flock -x "$mutex\_fd"

read\_count=$(cat "$read\_count\_file")

read\_count=$((read\_count + 1))

echo $read\_count > "$read\_count\_file"

if [ "$read\_count" -eq 1 ]; then

exec {rw\_mutex\_fd}>"$rw\_mutex\_file" # Lock rw\_mutex

flock -x "$rw\_mutex\_fd"

fi

flock -u "$mutex\_fd" # Unlock mutex

# Reading section

echo "Reader $(($1)) is reading: $(cat $shared\_data)"

sleep 2 # Simulate reading time

# Exit section

exec {mutex\_fd}>"$mutex\_file" # Lock mutex

flock -x "$mutex\_fd"

read\_count=$(cat "$read\_count\_file")

read\_count=$((read\_count - 1))

echo $read\_count > "$read\_count\_file"

if [ "$read\_count" -eq 0 ]; then

flock -u "$rw\_mutex\_fd" # Unlock rw\_mutex

fi

flock -u "$mutex\_fd" # Unlock mutex

sleep $((RANDOM % 3 + 1)) # Simulate time before next read

done

}

# Writer function

writer() {

while true; do

echo "Writer $(($1)) is trying to write..."

# Entry section

exec {rw\_mutex\_fd}>"$rw\_mutex\_file" # Lock rw\_mutex

flock -x "$rw\_mutex\_fd"

# Writing section

echo "Writer $(($1)) is writing data..."

echo "Data written by writer $(($1)) at $(date)" > "$shared\_data"

sleep 2 # Simulate writing time

# Exit section

flock -u "$rw\_mutex\_fd" # Unlock rw\_mutex

sleep $((RANDOM % 5 + 2)) # Simulate time before next write

done

}

# Create reader and writer processes

for i in {1..3}; do

reader $i &

done

for i in {1..2}; do

writer $i &

done

# Wait for all processes (infinite simulation)

Wait

OUTPUT:-

