Lab 5 Report

Tasks:

1) Identify the critical section and synchronize the program. You would need to use P(sem) and V(sem) operations at the right places to solve the synchronization problem. You will also need to find the logical flaw if there is any and try to solve it. To use P(sem) and V(sem) include the provided "sem.h" file. Your solution should solve the following problems:

- Prevent race conditions
- Prevent a son from withdrawing money when there is no balance.
- Prevent undefined outputs like negative balance
- Prevent a process from continuously requesting access to the shared memory. Prevent unnecessary cpu cycle.
- The problem should be solved with as few semaphore variables as possible.

2) Submit a report showing the critical section of the code (and logical errors) and explain your solution in detail.

Solution:

To solve this assignment, the dad process, first son process, and second son process were designated as critical portions. To prevent race conditions, there are sleep commands at the end of each process. With the addition of semaphores and each process running in order unnecessary CPU cycles are also prevented. To prevent a son process from withdrawing money when there isn't enough balance, a condition within their processes has been added that checks for this scenario. This also prevented undefined (negative) outputs from happening. Finally, I only used one semaphore, which acts as a flag that allows or prevents access to the shared memory. My solution can be viewed in the 'syncBank.c' file, where I have added comments showcasing my changes. In this report, I will be going over my specific changes on each critical portion.

In **line 42**, I changed the initBalance value from 0 to 100 because later in the code, at **line 66**, it is stated that the file 'balance.txt' should be initialized to be 100.

```
35  #define FALSE           1
36
37  #define NumOfDepositAttempt  5          /* How many times Dad will deposit money */
38  #define NumOfWithdrawAttempt 20         /* Total number of attempts SON_1 and SON_2 has */
39
40  #define DepositAmount        60        /* The amound of money Dad deposits at a time */
41  #define WithdrawAmount       20        /* The amount of money Son withdraws at a time */
42  #define initBalance          100       /// Changed the initial balance from $0 to $100 as stated in line 66
43
44  FILE *fp1, *fp2, *fp3, *fp4;      /* File Pointers */
```

```
66  //Initialize the file balance to be $100
67  fp1 = fopen("balance.txt","w+");
68  bal1 = initBalance; /// Logical error: the file starts with initBalance = 0 but the comment in line 66 states its 100
69                      /// so I changed initBalance to 100
70  fprintf(fp1, "%d\n", bal1);
71  fclose(fp1);
```

In **lines 56-64**, I added a semaphore that I will be used to give access to the shared memory to the current process. I followed a similar setup as in the syncExample.c file.

```
56    // ********************************************************************************
57    /// Setting the semaphores
58    /// Semaphores are used to allow and deny access to a common resource amongst
59    /// processes. In this case, that common resource is the balance in the dad's bank
60    /// account. So I created a semaphor that will be used to allow access to withdraw
61    /// money from the bank acount. I formatted it just like in syncExample.c
62    int semBank = semget(IPC_PRIVATE,1,0666 | IPC_CREAT);
63    sem_create(semBank, 1);
64    // ********************************************************************************
```

The first child process is the dad process (**lines 86-121**). In **lines 91-93**, the code enters the critical portion and so I called the P operator from the 'sem.h' file. Later in **line 103**, a sleep command is used so that the dad process sleeps and other processes can work in the meantime. To ensure that other processes have access to the shared memory during this time, I called the V operator from sem.h before the sleep command (**line 102**) to give other processes access, and a P operator after the sleep command (**line 104**) to give access back to the dad process. This will allow other processes to run and access the shared memory. Additionally, I read the balance.txt file one more time in case any changes to the balance have been made by the other processes while the dad process sleeps (**lines 106-107**). Finally, In **lines 107-109**, the dad process comes to an end and so I called the V operator from sem.h. I followed the same setup in 'syncExample.c' file as well as the comments in the 'sem.h' file

```
86    if (pid == CHILD) {
87    //First Child Process. Dad tries to do some updates.
88      printf("Dad's Pid: %d\n",getpid());
89      N = NumOfDepositAttempt;
90      for(i = 1; i <= N; i++) {
91        // ********************************************************************************
92        P(semBank); /// Calling P operation on semaphore at start of critical portion
93        // ********************************************************************************
94        printf("Dad is requesting to view the balance.\n"); //Dad is requesting to get hold of an ATM.
95        fp1 = fopen("balance.txt", "r+"); //Dad successfully got hold of the ATM.
96        fscanf(fp1, "%d", &bal2);
97        printf("Dad reads balance = %d \n", bal2);
98        int r = rand()%5 + 1;
99        printf("Dad needs %d sec to prepare money\n", r);
100
101       // ********************************************************************************
102       V(semBank); /// Calling V operation on semaphore to allow other processes to work
103       sleep(r); //Dad Process is sleeping for r sec. You need to make sure that other processes can work in the mean time.
104       P(semBank); /// Calling P operation on semaphore to allow other porcesses to work
105
106       fp1 = fopen("balance.txt", "r+"); /// Get new balance in case it was changed while dad Process prepared money
107       fscanf(fp1, "%d", &bal2);
108       // ********************************************************************************
109
110       fseek(fp1, 0L, 0); //Dad will now deposit the money. For this Dad will access the ATM again. And update the current balance.
111       bal2 += DepositAmount;
112       fprintf(fp1, "%d \n", bal2);
113       fclose(fp1);
114       printf("Dad writes new balance = %d \n", bal2);
115       printf("Dad will deposit %d more time\n", N-i); //Dad depostited the money.
116       // ********************************************************************************
117       V(semBank); /// Calling V operation on semaphore at end of critical portion
118       // ********************************************************************************
119       sleep(rand()%10 + 1); /* Dad will wait some time for requesting to see balance again.*/
120     }
121   }
```

The second child process is the first son process (**lines 130-177**). Just like the dad process, I called the P operator at the beginning of the critical portion of the first son process (**lines 135-137**). Later in **lines 152-174**, the first son process ends and I call the V operator. To ensure that the first son process does not withdraw money if there isn't enough to withdraw, I added an if check in **line 152-163** that will check if the current balance is more than the withdrawAmount. If this is the case then the son will withdraw money. Otherwise, a message stating that there isn't enough balance will display.

```
123   else {
124     //Parent Process. Fork off another child process.
125     if ((pid = fork()) == -1) {
126       //Fork failed!
127       perror("fork");
128       exit(1);
129     }
130     if (pid == CHILD) {
131       printf("First Son's Pid: %d\n",getpid());
132       //Second child process. First poor son tries to do updates.
133       flag = FALSE;
134       while(flag == FALSE) {
135         // ********************************************************************************
136         P(semBank); /// Calling P operation on semaphore at start of critical portion
137         // ********************************************************************************
138         printf("SON_1 is requesting to view the balance.\n"); //Son_1 is requesting to get hold of the ATM.
139         fp3 = fopen("attempt.txt" , "r+"); //Son_1 successfully got hold of the ATM.
140         fscanf(fp3, "%d", &N_Att); // Son_1 Checks if he has more than 0 attempt remaining.
141         printf("Attempt remaining: %d.\n", N_Att);
142         if(N_Att == 0) {
143           fclose(fp3);
144           flag = TRUE;
145         }
146         else {
147           fp2 = fopen("balance.txt", "r+");//Son_1 reads the balance.
148           fscanf(fp2,"%d", &bal2);
149           printf("SON_1 reads balance. Available Balance: %d \n", bal2);
150           printf("SON_1 wants to withdraw money. "); //And if balance is greater than Withdraw amount, then son can withdraw money.
151
152           // ********************************************************************************
153           /// Check if there is enough balance for son 1 to withdraw $20
154           if (bal2 >= WithdrawAmount) { /// if there is enough balance, withdraw money
155             fseek(fp2,0L, 0);
156             bal2 -=WithdrawAmount;
157             fprintf(fp2,"%d\n", bal2);
158             fclose(fp2);
159             printf("SON_1 withdrawed %d. New Balance: %d \n",WithdrawAmount, bal2);
160           } else { /// If not enough balance, state there isn't enough balance
161             printf("\nNot enough balance to withdraw\n");
162           }
163           // ********************************************************************************
164
165           fseek(fp3,0L, 0); //SON_1 will write the number of  attempt remaining in the attampt.txt file.
166           N_Att -=1;
167           fprintf(fp3, "%d\n", N_Att);
168           fclose(fp3);
169           printf("Number of attempts remaining:%d \n", N_Att);
170
171         }
172         // ********************************************************************************
173         V(semBank); /// Calling V operation on semaphore at end of critical portion
174         // ********************************************************************************
175         sleep(rand()%10+1); //SON_1 will wait some time before the next request.
176       }
177     }
```

The third child process is the second son process (**lines 186-234)**. Just like the previous processes, I called the P operator at the beginning of the second son process in **lines 191-193**. Later in **lines 229-231**, the second son process ends and I call the V operator. To ensure that the second son process does not withdraw money if there isn't enough to withdraw, I added an if check in **line 209-220** just like in the first son process.

```
180     //Parent Process. Fork off one more child process.
181       if ((pid = fork()) == -1) {
182         //fork failed!
183         perror("fork");
184         exit(1);
185       }
186       if (pid == CHILD) {
187         printf("Second Son's Pid: %d\n",getpid());
188         //Third child process. Second poor son tries to do updates.
189         flag1 = FALSE;
190         while(flag1 == FALSE) {
191           // ********************************************************************************
192           P(semBank); /// Calling P operation on semaphore at start of critical portion
193           // ********************************************************************************
194           printf("SON_2 is requesting to view the balance.\n"); //Son_2 is requesting to get hold of the ATM.
195           fp3 = fopen("attempt.txt" , "r+"); //Son_2 successfully got hold of the ATM.
196           fscanf(fp3, "%d", &N_Att); // Son_2 Checks if he has more than 0 attempt remaining.
197           printf("Attempt remaining: %d.\n", N_Att);
198           if(N_Att == 0) {
199             fclose(fp3);
200             flag1 = TRUE;
201           }
202           else {
203
204             fp2 = fopen("balance.txt", "r+"); //Son_2 reads the balance.
205             fscanf(fp2,"%d", &bal2);
206             printf("SON_2 reads balance. Available Balance: %d \n", bal2);
207             printf("SON_2 wants to withdraw money. "); //And if balance is greater than Withdraw amount, then son can withdraw money.
208
209             // ********************************************************************************
210             /// Check if there is enough balance for son 2 to withdraw $20
211             if (bal2 >= WithdrawAmount) { /// if there is enough balance, withdraw money
212               fseek(fp2,0L, 0);
213               bal2 -=WithdrawAmount;
214               fprintf(fp2,"%d\n", bal2);
215               fclose(fp2);
216               printf("SON_2 withdrawed %d. New Balance: %d \n",WithdrawAmount, bal2);
217             } else { /// If not enough balance, state there isn't enough balance
218               printf("Not enough balance to withdraw\n");
219             }
220             // ********************************************************************************
221
222             fseek(fp3,0L, 0); //SON_2 will write the number of  attempt remaining in the attampt.txt file.
223             N_Att -=1;
224             fprintf(fp3, "%d\n", N_Att);
225             fclose(fp3);
226             printf("Number of attempts remaining: %d \n", N_Att);
227
228           }
229           // ********************************************************************************
230           V(semBank); /// Calling V operation on semaphore at end of critical portion
231           // ********************************************************************************
232           sleep(rand()%10+1);//SON_2 will wait some time before the next request.
233         }
234       }
```

Output:

A sample output can be fully seen in the ‘Test output.txt’ file

```
Dad's Pid: 38438
Dad is requesting to view the balance.
Dad reads balance = 100
Dad needs 2 sec to prepare money
First Son's Pid: 38439
SON_1 is requesting to view the balance.
Second Son's Pid: 38440
Attempt remaining: 20.
SON_1 reads balance. Available Balance: 100
SON_1 wants to withdraw money. SON_1 withdrawed 20. New Balance: 80
Number of attempts remaining:19
SON_2 is requesting to view the balance.
Attempt remaining: 19.
SON_2 reads balance. Available Balance: 80
SON_2 wants to withdraw money. SON_2 withdrawed 20. New Balance: 60
Number of attempts remaining: 18
Dad writes new balance = 120
Dad will deposit 4 more time
SON_1 is requesting to view the balance.
Attempt remaining: 18.
SON_1 reads balance. Available Balance: 120
SON_1 wants to withdraw money. SON_1 withdrawed 20. New Balance: 100
Number of attempts remaining:17
SON_2 is requesting to view the balance.
Attempt remaining: 17.
SON_2 reads balance. Available Balance: 100
SON_2 wants to withdraw money. SON_2 withdrawed 20. New Balance: 80
Number of attempts remaining: 16
Dad is requesting to view the balance.
Dad reads balance = 80
Dad needs 3 sec to prepare money
Dad writes new balance = 140
Dad will deposit 3 more time
SON_1 is requesting to view the balance.
Attempt remaining: 16.
SON_1 reads balance. Available Balance: 140
SON_1 wants to withdraw money. SON_1 withdrawed 20. New Balance: 120
Number of attempts remaining:15
SON_2 is requesting to view the balance.
Attempt remaining: 15.
SON_2 reads balance. Available Balance: 120
SON_2 wants to withdraw money. SON_2 withdrawed 20. New Balance: 100
Number of attempts remaining: 14
Dad is requesting to view the balance.
Dad reads balance = 100
```

```
Dad needs 4 sec to prepare money
SON_1 is requesting to view the balance.
Attempt remaining: 14.
SON_1 reads balance. Available Balance: 100
SON_1 wants to withdraw money. SON_1 withdrawed 20. New Balance: 80
Number of attempts remaining:13
SON_2 is requesting to view the balance.
Attempt remaining: 13.
SON_2 reads balance. Available Balance: 80
SON_2 wants to withdraw money. SON_2 withdrawed 20. New Balance: 60
Number of attempts remaining: 12
Dad writes new balance = 120
Dad will deposit 2 more time
SON_2 is requesting to view the balance.
Attempt remaining: 12.
SON_2 reads balance. Available Balance: 120
SON_2 wants to withdraw money. SON_2 withdrawed 20. New Balance: 100
Number of attempts remaining: 11
SON_1 is requesting to view the balance.
Attempt remaining: 11.
SON_1 reads balance. Available Balance: 100
SON_1 wants to withdraw money. SON_1 withdrawed 20. New Balance: 80
Number of attempts remaining:10
Dad is requesting to view the balance.
Dad reads balance = 80
Dad needs 3 sec to prepare money
Dad writes new balance = 140
Dad will deposit 1 more time
SON_2 is requesting to view the balance.
Attempt remaining: 10.
SON_2 reads balance. Available Balance: 140
SON_2 wants to withdraw money. SON_2 withdrawed 20. New Balance: 120
Number of attempts remaining: 9
SON_1 is requesting to view the balance.
Attempt remaining: 9.
SON_1 reads balance. Available Balance: 120
SON_1 wants to withdraw money. SON_1 withdrawed 20. New Balance: 100
Number of attempts remaining:8
Dad is requesting to view the balance.
Dad reads balance = 100
Dad needs 2 sec to prepare money
Dad writes new balance = 160
Dad will deposit 0 more time
child(pid = 38438) exited with the status 0.
SON_2 is requesting to view the balance.
```

```
Attempt remaining: 8.
SON_2 reads balance. Available Balance: 160
SON_2 wants to withdraw money. SON_2 withdrawed 20. New Balance: 140
Number of attempts remaining: 7
SON_1 is requesting to view the balance.
Attempt remaining: 7.
SON_1 reads balance. Available Balance: 140
SON_1 wants to withdraw money. SON_1 withdrawed 20. New Balance: 120
Number of attempts remaining:6
SON_2 is requesting to view the balance.
Attempt remaining: 6.
SON_2 reads balance. Available Balance: 120
SON_2 wants to withdraw money. SON_2 withdrawed 20. New Balance: 100
Number of attempts remaining: 5
SON_1 is requesting to view the balance.
Attempt remaining: 5.
SON_1 reads balance. Available Balance: 100
SON_1 wants to withdraw money. SON_1 withdrawed 20. New Balance: 80
Number of attempts remaining:4
SON_2 is requesting to view the balance.
Attempt remaining: 4.
SON_2 reads balance. Available Balance: 80
SON_2 wants to withdraw money. SON_2 withdrawed 20. New Balance: 60
Number of attempts remaining: 3
SON_1 is requesting to view the balance.
Attempt remaining: 3.
SON_1 reads balance. Available Balance: 60
SON_1 wants to withdraw money. SON_1 withdrawed 20. New Balance: 40
Number of attempts remaining:2
SON_2 is requesting to view the balance.
Attempt remaining: 2.
SON_2 reads balance. Available Balance: 40
SON_2 wants to withdraw money. SON_2 withdrawed 20. New Balance: 20
Number of attempts remaining: 1
SON_1 is requesting to view the balance.
Attempt remaining: 1.
SON_1 reads balance. Available Balance: 20
SON_1 wants to withdraw money. SON_1 withdrawed 20. New Balance: 0
Number of attempts remaining:0
SON_2 is requesting to view the balance.
Attempt remaining: 0.
SON_1 is requesting to view the balance.
Attempt remaining: 0.
child(pid = 38440) exited with the status 0.
child(pid = 38439) exited with the status 0.
```