Assignment 12 | Operating System CE-092

Assignment submission for Operating System subject week 12.

nevilparmar24@gmail.com

Task 1:

Banker's algorithm.

Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <pthread.h>
#include <stdbool.h>

int nResources,
    nProcesses;
int *resources;
int **allocated;
int **maxRequired;
int **need;
int *safeSeq;
int nProcessRan = 0;

pthread_mutex_t lockResources;
pthread_cond_t condition;

bool getSafeSeq();
```

```
void* processCode(void* );

int main(int argc, char** argv) {
    srand(time(NULL));

    printf("\nNumber of processes? ");
    scanf("%d", &nProcesses);

    printf("\nNumber of resources? ");
    scanf("%d", &nResources);

    resources = (int *)malloc(nResources *
sizeof(*resources));
    printf("\nCurrently Available resources (R1 R2
...)? ");
    for (int i = 0; i < nResources; i++)
        scanf("%d", &resources[i]);

    allocated = (int **)malloc(nProcesses *
sizeof(*allocated));
    for (int i = 0; i < nProcesses; i++)
        allocated[i] = (int *)malloc(nResources *
sizeof(**allocated));

    maxRequired = (int **)malloc(nProcesses *
sizeof(*maxRequired));
    for (int i = 0; i < nProcesses; i++)
        maxRequired[i] = (int *)malloc(nResources *
sizeof(**maxRequired));

    // allocated
    printf("\n");
    for (int i = 0; i < nProcesses; i++) {
        printf("\nResource allocated to process %d (R1
```

```
R2 ...)? ", i + 1);
        for (int j = 0; j < nResources; j++)
            scanf("%d", &allocated[i][j]);
    }
    printf("\n");

    // maximum required resources
    for (int i = 0; i < nProcesses; i++) {
        printf("\nMaximum resource required by process
%d (R1 R2 ...)? ", i + 1);
        for (int j = 0; j < nResources; j++)
            scanf("%d", &maxRequired[i][j]);
    }
    printf("\n");

    // calculate need matrix
    need = (int **)malloc(nProcesses * sizeof(*need));
    for (int i = 0; i < nProcesses; i++)
        need[i] = (int *)malloc(nResources *
sizeof(**need));

    for (int i = 0; i < nProcesses; i++)
        for (int j = 0; j < nResources; j++)
            need[i][j] = maxRequired[i][j] -
allocated[i][j];

    // get safe sequence
    safeSeq = (int *)malloc(nProcesses *
sizeof(*safeSeq));
    for (int i = 0; i < nProcesses; i++) safeSeq[i] =
-1;

    if (!getSafeSeq()) {
        printf("\nUnsafe State! The processes leads the
```

```
system to a unsafe state.\n\n");
        exit(-1);
    }

    printf("\n\nSafe Sequence Found : ");
    for (int i = 0; i < nProcesses; i++) {
        printf("%-3d", safeSeq[i] + 1);
    }

    printf("\nExecuting Processes...\n\n");
    sleep(1);

    // run threads
    pthread_t processes[nProcesses];
    pthread_attr_t attr;
    pthread_attr_init(&attr);

    int processNumber[nProcesses];
    for (int i = 0; i < nProcesses; i++)
processNumber[i] = i;

    for (int i = 0; i < nProcesses; i++)
        pthread_create(&processes[i], &attr,
processCode, (void *)(&processNumber[i]));

    for (int i = 0; i < nProcesses; i++)
        pthread_join(processes[i], NULL);

    printf("\nAll Processes Finished\n");

    // free resources
    free(resources);
    for (int i = 0; i < nProcesses; i++) {
        free(allocated[i]);
```

```
        free(maxRequired[i]);
        free(need[i]);
    }
    free(allocated);
    free(maxRequired);
    free(need);
    free(safeSeq);
}


bool getSafeSeq() {

    int tempRes[nResources];
    for (int i = 0; i < nResources; i++) tempRes[i] =
resources[i];

    bool finished[nProcesses];
    for (int i = 0; i < nProcesses; i++) finished[i] =
false;
    int nfinished = 0;
    while (nfinished < nProcesses) {
        bool safe = false;

        for (int i = 0; i < nProcesses; i++) {
            if (!finished[i]) {
                bool possible = true;

                for (int j = 0; j < nResources; j++)
                    if (need[i][j] > tempRes[j]) {
                        possible = false;
                        break;
                    }

                if (possible) {
```

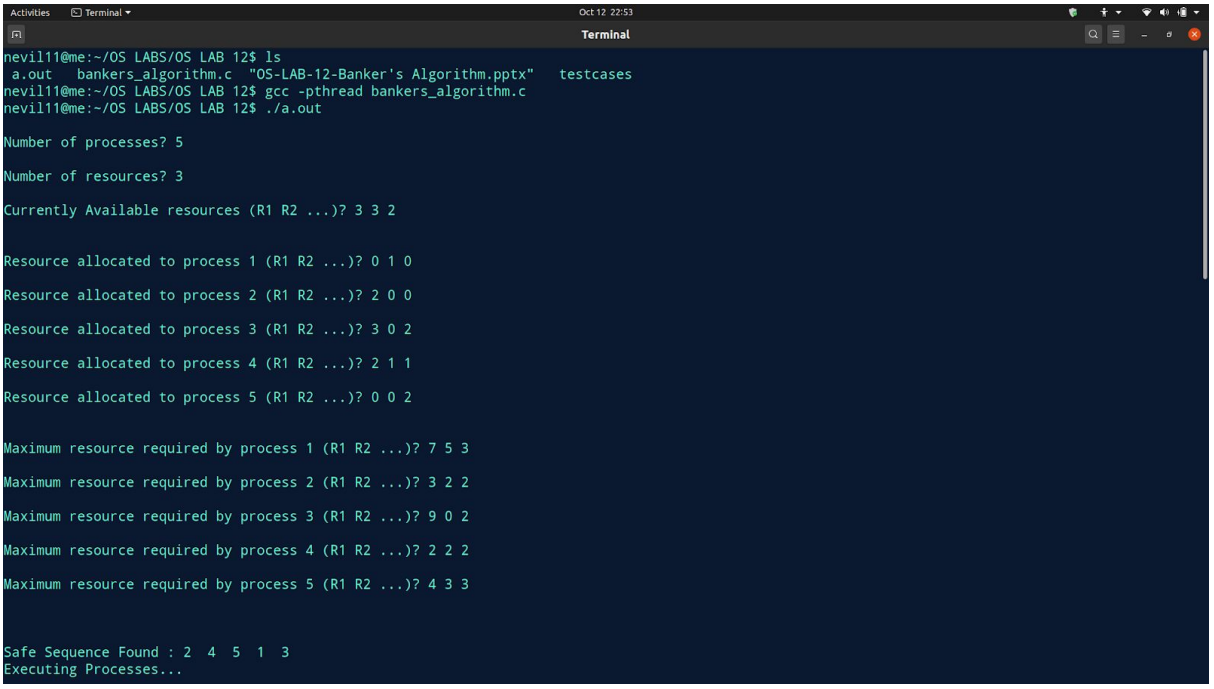
```
                    for (int j = 0; j < nResources;
j++)
                        tempRes[j] += allocated[i][j];
                    safeSeq[nfinished] = i;
                    finished[i] = true;
                    ++nfinished;
                    safe = true;
                }
            }
        }

        if (!safe) {
            for (int k = 0; k < nProcesses; k++)
safeSeq[k] = -1;
            return false; // no safe sequence found
        }
    }
    return true; // safe sequence found
}

// function to simulate the behaviour of resource
allocation to any process
void* processCode(void *arg) {
    int p = *((int *) arg);

    // lock resources
    pthread_mutex_lock(&lockResources);

    // condition check
    while (p != safeSeq[nProcessRan])
        pthread_cond_wait(&condition, &lockResources);

    // process
    printf("\n--> Process %d", p + 1);
```

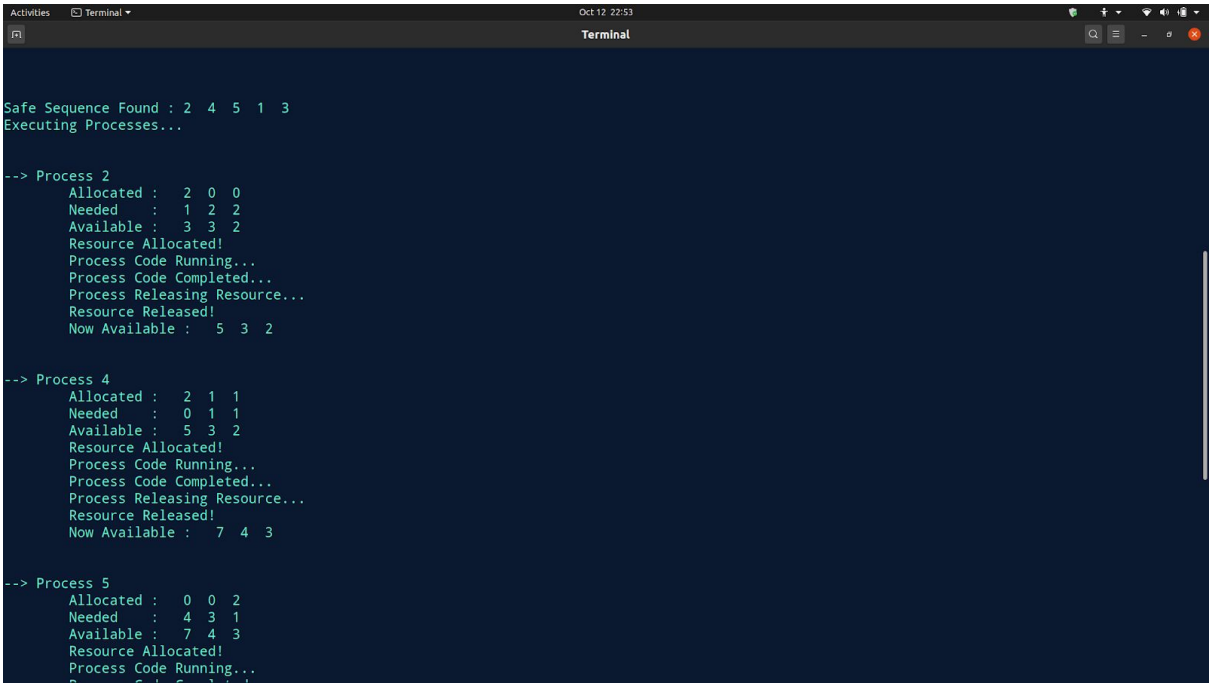
```
    printf("\n\tAllocated : ");
    for (int i = 0; i < nResources; i++)
        printf("%3d", allocated[p][i]);

    printf("\n\tNeeded    : ");
    for (int i = 0; i < nResources; i++)
        printf("%3d", need[p][i]);

    printf("\n\tAvailable : ");
    for (int i = 0; i < nResources; i++)
        printf("%3d", resources[i]);

    printf("\n"); sleep(1);

    printf("\tResource Allocated!");
    printf("\n"); sleep(1);
    printf("\tProcess Code Running...");
    printf("\n"); sleep(rand() % 3 + 2); // process
code
    printf("\tProcess Code Completed...");
    printf("\n"); sleep(1);
    printf("\tProcess Releasing Resource...");
    printf("\n"); sleep(1);
    printf("\tResource Released!");

    for (int i = 0; i < nResources; i++)
        resources[i] += allocated[p][i];

    printf("\n\tNow Available : ");
    for (int i = 0; i < nResources; i++)
        printf("%3d", resources[i]);
    printf("\n\n");

    sleep(1);
```

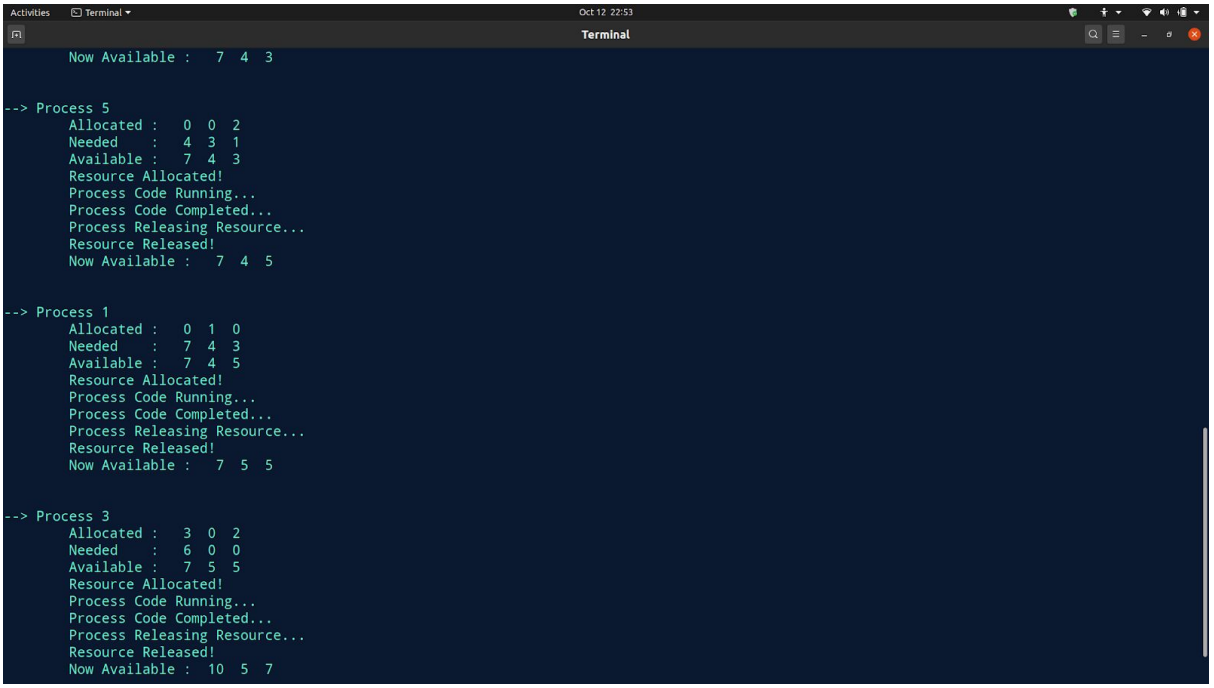
```
    // condition broadcast
    nProcessRan++;
    pthread_cond_broadcast(&condition);
    pthread_mutex_unlock(&lockResources);
    pthread_exit(NULL);
}
```

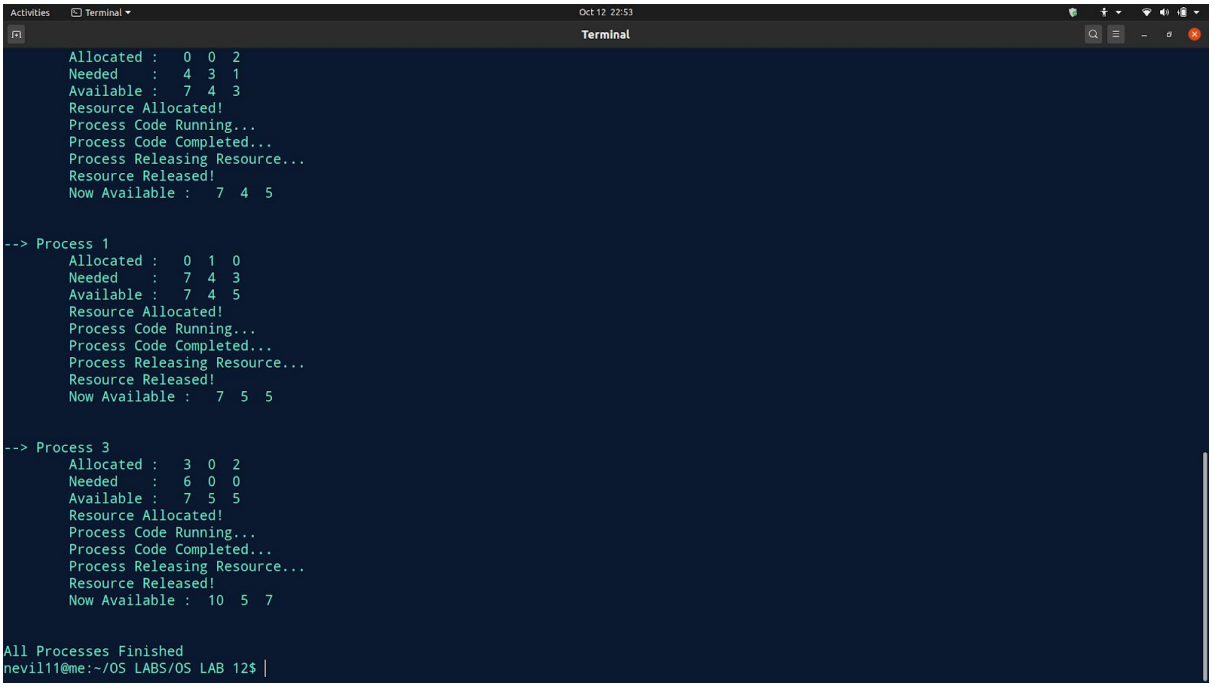
Output:

Safe sequence demo.

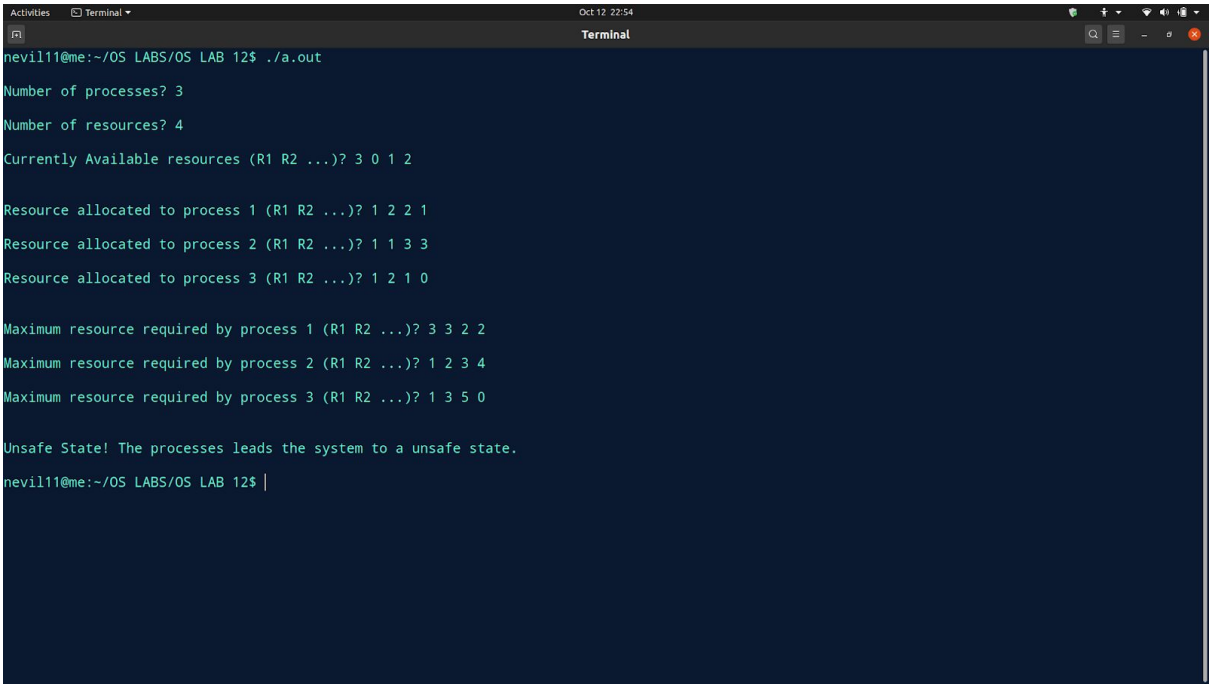



Activities
Terminal
Oct 12 22:53
Terminal
Safe Sequence Found : 2 4 5 1 3
Executing Processes...
--> Process 2
Allocated : 2 0 0
Needed : 1 2 2
Available : 3 3 2
Resource Allocated!
Process Code Running...
Process Code Completed...
Process Releasing Resource...
Resource Released!
Now Available : 5 3 2
--> Process 4
Allocated : 2 1 1
Needed : 0 1 1
Available : 5 3 2
Resource Allocated!
Process Code Running...
Process Code Completed...
Process Releasing Resource...
Resource Released!
Now Available : 7 4 3
--> Process 5
Allocated : 0 0 2
Needed : 4 3 1
Available : 7 4 3
Resource Allocated!
Process Code Running...


Activities
Terminal
Oct 12 22:53
Terminal
Now Available : 7 4 3
--> Process 5
Allocated : 0 0 2
Needed : 4 3 1
Available : 7 4 3
Resource Allocated!
Process Code Running...
Process Code Completed...
Process Releasing Resource...
Resource Released!
Now Available : 7 4 5
--> Process 1
Allocated : 0 1 0
Needed : 7 4 3
Available : 7 4 5
Resource Allocated!
Process Code Running...
Process Code Completed...
Process Releasing Resource...
Resource Released!
Now Available : 7 5 5
--> Process 3
Allocated : 3 0 2
Needed : 6 0 0
Available : 7 5 5
Resource Allocated!
Process Code Running...
Process Code Completed...
Process Releasing Resource...
Resource Released!
Now Available : 10 5 7



Unsafe sequence demo.



Nevil Parmar
CE-092
https://nevilparmar.me