CSE 321: Operating Systems

Lab Assignment 4

Name: Mollah Md Saif

ID: 20101416

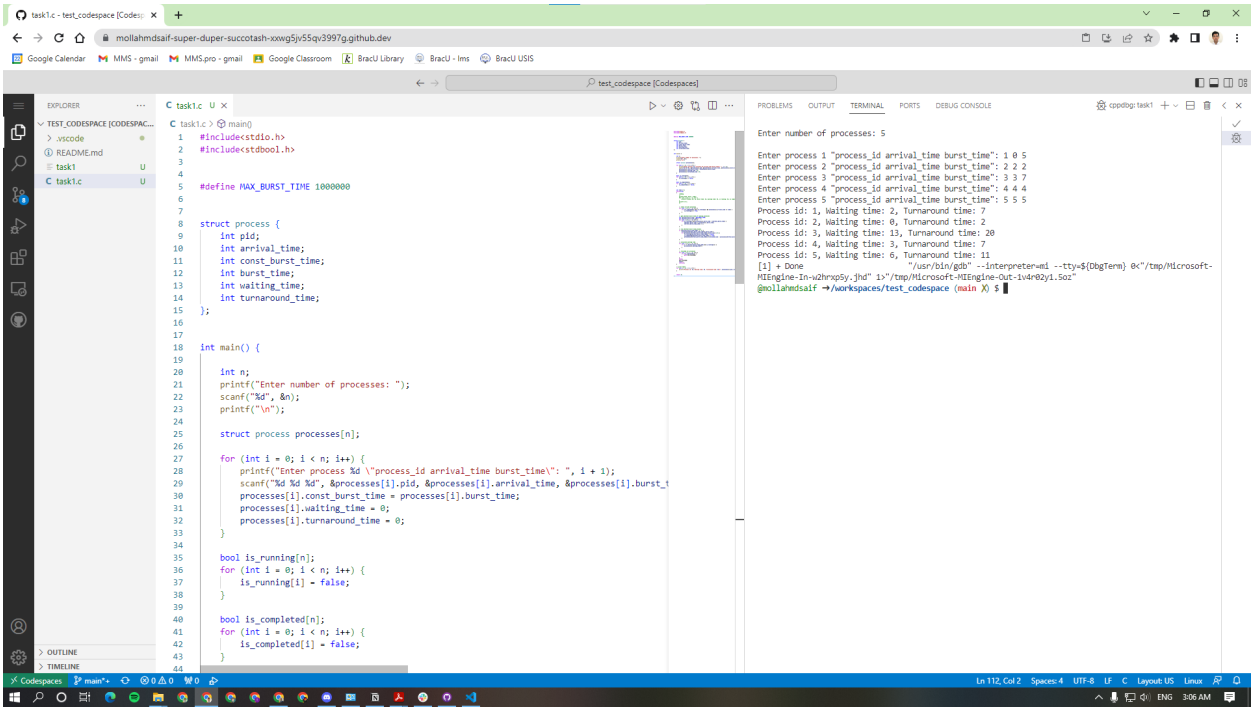
Section: 03

Task 1

```
#include<stdio.h>
#include<stdbool.h>

#define MAX_BURST_TIME 1000000

struct process {
    int pid;
    int arrival_time;
    int const_burst_time;
    int burst_time;
    int waiting_time;
    int turnaround_time;
};

int main() {

    int n;
    printf("Enter number of processes: ");
    scanf("%d", &n);
    printf("\n");

    struct process processes[n];

    for (int i = 0; i < n; i++) {
        printf("Enter process %d \"process_id arrival_time burst_time\": ", i + 1);
        scanf("%d %d %d", &processes[i].pid, &processes[i].arrival_time, &processes[i].burst_time);
        processes[i].const_burst_time = processes[i].burst_time;
        processes[i].waiting_time = 0;
        processes[i].turnaround_time = 0;
    }

    bool is_running[n];
    for (int i = 0; i < n; i++) {
        is_running[i] = false;
    }
```

```
    bool is_completed[n];
    for (int i = 0; i < n; i++) {
        is_completed[i] = false;
    }



    int time = 0;
    while(true)
    {
        //debug
        /*
        printf("Time: %d\n", time);
        for (int i = 0; i < n; i++) {
            printf("Process id: %d, burst time: %d, waiting time: %d, is running: %d, is completed: %d\n",
processes[i].pid, processes[i].burst_time, processes[i].waiting_time, is_running[i], is_completed[i]);
        }
        printf("\n");
        */



        // check arrived processes
        for (int i = 0; i < n; i++) {
            if (!is_completed[i] && !is_running[i] && processes[i].arrival_time <= time) {
                is_running[i] = true;
            }
        }


        // get shortest burst time of running processes
        int shortest_burst_time = MAX_BURST_TIME;
        int shortest_burst_time_index = -1;
        for (int i = 0; i < n; i++) {
            if (is_running[i] && processes[i].burst_time < shortest_burst_time) {
                shortest_burst_time = processes[i].burst_time;
                shortest_burst_time_index = i;
            }
        }


        // run shortest burst time process
```

```
        if (shortest_burst_time_index != -1) {
            processes[shortest_burst_time_index].burst_time--;
            if (processes[shortest_burst_time_index].burst_time == 0) {
                is_running[shortest_burst_time_index] = false;
                is_completed[shortest_burst_time_index] = true;
                processes[shortest_burst_time_index].turnaround_time =
processes[shortest_burst_time_index].waiting_time + processes[shortest_burst_time_index].const_burst_time;
            }
        }

        // calculate waiting time
        for (int i = 0; i < n; i++) {
            if (i != shortest_burst_time_index && is_running[i]) {
                processes[i].waiting_time++;
            }
        }

        // finished all processes
        for (int i = 0; i < n; i++) {
            if (!is_completed[i]) {
                goto not_finished;
            }
        }
        break;
        not_finished:
        time++;
        continue;
    }

    // print table
    for (int i = 0; i < n; i++) {
        printf("Process id: %d, Waiting time: %d, Turnaround time: %d\n", processes[i].pid,
processes[i].waiting_time, processes[i].turnaround_time);
    }

    return 0;
}
```

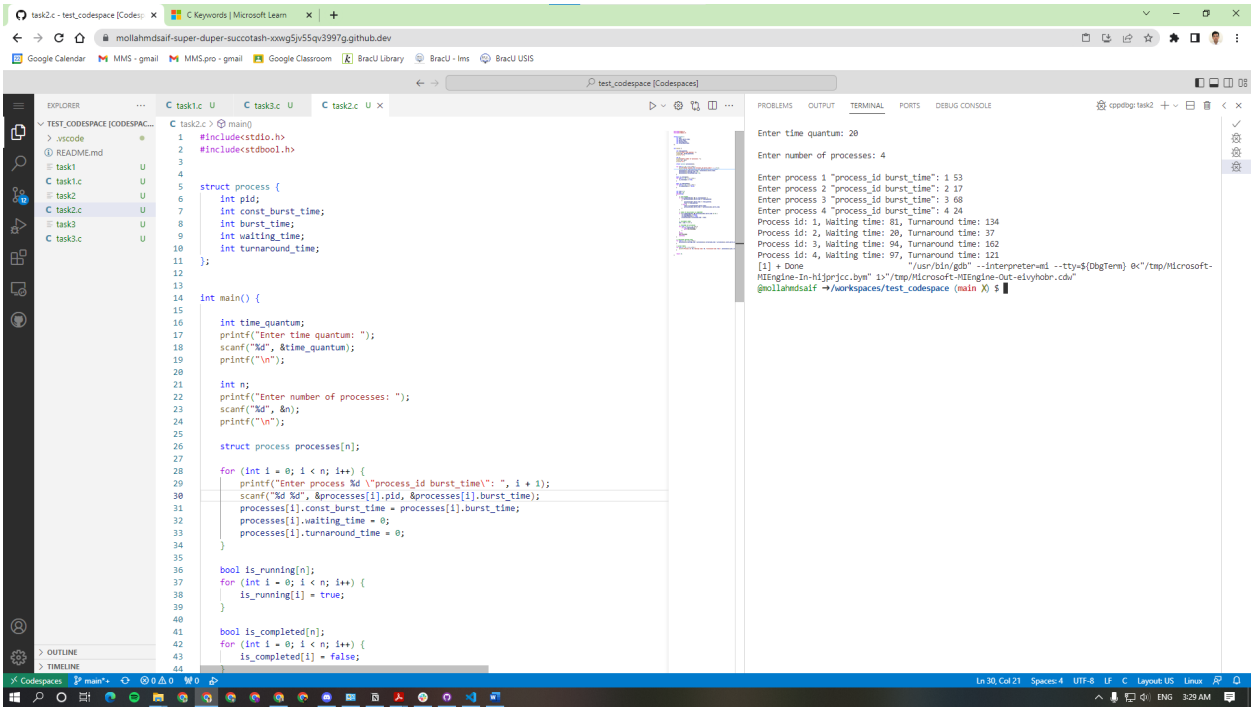

task1.c - test_codespace [Codespaces]
mollahmdsaif-super-duper-succotash-xxwg5jv55qv3997g.github.dev
test_codespace [Codespaces]
EXPLORER
TEST_CODESPACE [CODESPAC...
.vscode
README.md
task1
task1.c
OUTLINE
TIMELINE
task1.c U
task1.c > main()
#include<stdio.h>
#include<stdbool.h>
#define MAX_BURST_TIME 1000000
struct process {
int pid;
int arrival_time;
int const_burst_time;
int burst_time;
int waiting_time;
int turnaround_time;
};
int main() {
int n;
printf("Enter number of processes: ");
scanf("%d", &n);
printf("\n");
struct process processes[n];
for (int i = 0; i < n; i++) {
printf("Enter process %d \"process_id arrival_time burst_time\": ", i + 1);
scanf("%d %d %d", &processes[i].pid, &processes[i].arrival_time, &processes[i].burst_t
processes[i].const_burst_time = processes[i].burst_time;
processes[i].waiting_time = 0;
processes[i].turnaround_time = 0;
}
bool is_running[n];
for (int i = 0; i < n; i++) {
is_running[i] = false;
}
bool is_completed[n];
for (int i = 0; i < n; i++) {
is_completed[i] = false;
}
PROBLEMS
OUTPUT
TERMINAL
PORTS
DEBUG CONSOLE
cppdbg: task1
Enter number of processes: 5
Enter process 1 "process_id arrival_time burst_time": 1 0 5
Enter process 2 "process_id arrival_time burst_time": 2 2 2
Enter process 3 "process_id arrival_time burst_time": 3 3 7
Enter process 4 "process_id arrival_time burst_time": 4 4 4
Enter process 5 "process_id arrival_time burst_time": 5 5 5
Process id: 1, Waiting time: 2, Turnaround time: 7
Process id: 2, Waiting time: 0, Turnaround time: 2
Process id: 3, Waiting time: 13, Turnaround time: 20
Process id: 4, Waiting time: 3, Turnaround time: 7
Process id: 5, Waiting time: 6, Turnaround time: 11
[1] + Done "/usr/bin/gdb" --interpreter=mi --tty=${DbgTerm} 0<"/tmp/Microsoft-MIEngine-In-w2hrxp5y.jhd" 1>"/tmp/Microsoft-MIEngine-Out-1v4r02y1.5oz"
@mollahmdsaif ➜/workspaces/test_codespace (main X) $
Codespaces
main*+
Ln 112, Col 2
Spaces: 4
UTF-8
LF
C
Layout: US
Linux
ENG
3:06 AM

Task 2

```
#include<stdio.h>
#include<stdbool.h>

struct process {
    int pid;
    int const_burst_time;
    int burst_time;
    int waiting_time;
    int turnaround_time;
};

int main() {

    int time_quantum;
    printf("Enter time quantum: ");
    scanf("%d", &time_quantum);
    printf("\n");

    int n;
    printf("Enter number of processes: ");
    scanf("%d", &n);
    printf("\n");

    struct process processes[n];

    for (int i = 0; i < n; i++) {
        printf("Enter process %d \"process_id burst_time\": ", i + 1);
        scanf("%d %d", &processes[i].pid, &processes[i].burst_time);
        processes[i].const_burst_time = processes[i].burst_time;
        processes[i].waiting_time = 0;
        processes[i].turnaround_time = 0;
    }

    bool is_running[n];
```

```
for (int i = 0; i < n; i++) {
    is_running[i] = true;
}

bool is_completed[n];
for (int i = 0; i < n; i++) {
    is_completed[i] = false;
}



int time = 0;
int idx = 0;
while(true)
{
    // run process
    if (!is_completed[idx] && is_running[idx]) {
        if (processes[idx].burst_time >= time_quantum)
        {
            processes[idx].burst_time -= time_quantum;
            time += time_quantum;
        } else {
            time += processes[idx].burst_time;
            processes[idx].burst_time -= processes[idx].burst_time;
        }
    }

    // check if any process is completed
    if (!is_completed[idx] && processes[idx].burst_time == 0) {
        is_running[idx] = false;
        is_completed[idx] = true;
        processes[idx].turnaround_time = time;
    }

    // increment index
    idx = (idx + 1) % n;
```

```
        // finished all processes
        for (int i = 0; i < n; i++) {
            if (!is_completed[i]) {
                goto not_finished;
            }
        }
        break;
        not_finished:
        continue;
    }

    // calculate waiting time
    for (int i = 0; i < n; i++) {
        processes[i].waiting_time = processes[i].turnaround_time - processes[i].const_burst_time;
    }

    // print table
    for (int i = 0; i < n; i++) {
        printf("Process id: %d, Waiting time: %d, Turnaround time: %d\n", processes[i].pid,
processes[i].waiting_time, processes[i].turnaround_time);
    }



    return 0;
}
```


task2.c - test_codespace [Codespaces]
C Keywords | Microsoft Learn
mollahmdsaif-super-duper-succotash-xxwg5jv55qv3997g.github.dev
Google Calendar
MMS - gmail
MMS.pro - gmail
Google Classroom
BracU Library
BracU - lms
BracU USIS
test_codespace [Codespaces]
EXPLORER
TEST_CODESPACE [CODESPAC...
.vscode
README.md
task1
task1.c
task2
task2.c
task3
task3.c
OUTLINE
TIMELINE
task1.c U
task3.c U
task2.c U
task2.c > main()
#include<stdio.h>
#include<stdbool.h>
struct process {
int pid;
int const_burst_time;
int burst_time;
int waiting_time;
int turnaround_time;
};
int main() {
int time_quantum;
printf("Enter time quantum: ");
scanf("%d", &time_quantum);
printf("\n");
int n;
printf("Enter number of processes: ");
scanf("%d", &n);
printf("\n");
struct process processes[n];
for (int i = 0; i < n; i++) {
printf("Enter process %d \"process_id burst_time\": ", i + 1);
scanf("%d %d", &processes[i].pid, &processes[i].burst_time);
processes[i].const_burst_time = processes[i].burst_time;
processes[i].waiting_time = 0;
processes[i].turnaround_time = 0;
}
bool is_running[n];
for (int i = 0; i < n; i++) {
is_running[i] = true;
}
bool is_completed[n];
for (int i = 0; i < n; i++) {
is_completed[i] = false;
}
PROBLEMS
OUTPUT
TERMINAL
PORTS
DEBUG CONSOLE
cppdbg: task2
Enter time quantum: 20
Enter number of processes: 4
Enter process 1 "process_id burst_time": 1 53
Enter process 2 "process_id burst_time": 2 17
Enter process 3 "process_id burst_time": 3 68
Enter process 4 "process_id burst_time": 4 24
Process id: 1, Waiting time: 81, Turnaround time: 134
Process id: 2, Waiting time: 20, Turnaround time: 37
Process id: 3, Waiting time: 94, Turnaround time: 162
Process id: 4, Waiting time: 97, Turnaround time: 121
[1] + Done "/usr/bin/gdb" --interpreter=mi --tty=${DbgTerm} 0<"/tmp/Microsoft-MIEngine-In-hijprjcc.bym" 1>"/tmp/Microsoft-MIEngine-Out-eivyhobr.cdv"
@mollahmdsaif ➜ /workspaces/test_codespace (main X) $
Codespaces
main*+
Ln 30, Col 21
Spaces: 4
UTF-8
LF
C
Layout: US
Linux
ENG
3:29 AM

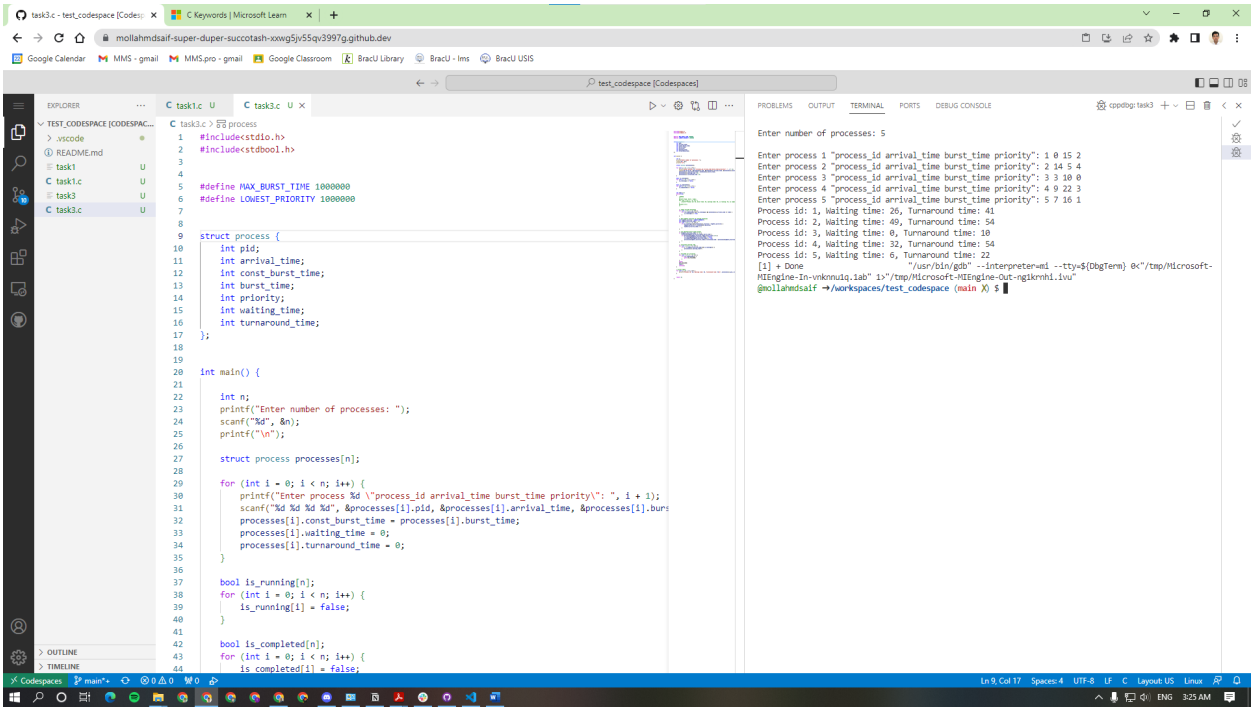
Task 3

```
#include<stdio.h>

#include<stdbool.h>



#define MAX_BURST_TIME 1000000

#define LOWEST_PRIORITY 1000000



struct process {

    int pid;

    int arrival_time;

    int const_burst_time;

    int burst_time;

    int priority;

    int waiting_time;

    int turnaround_time;

};



int main() {


    int n;

    printf("Enter number of processes: ");

    scanf("%d", &n);

    printf("\n");


    struct process processes[n];


    for (int i = 0; i < n; i++) {

        printf("Enter process %d \"process_id arrival_time burst_time priority\": ", i + 1);

        scanf("%d %d %d %d", &processes[i].pid, &processes[i].arrival_time, &processes[i].burst_time,
&processes[i].priority);

        processes[i].const_burst_time = processes[i].burst_time;

        processes[i].waiting_time = 0;

        processes[i].turnaround_time = 0;
```

```
    }

    bool is_running[n];
    for (int i = 0; i < n; i++) {
        is_running[i] = false;
    }

    bool is_completed[n];
    for (int i = 0; i < n; i++) {
        is_completed[i] = false;
    }


    int time = 0;
    while(true)
    {
        //debug
        /*
        printf("Time: %d\n", time);
        for (int i = 0; i < n; i++) {
            printf("Process id: %d, burst time: %d, waiting time: %d, is running: %d, is completed: %d\n",
processes[i].pid, processes[i].burst_time, processes[i].waiting_time, is_running[i], is_completed[i]);
        }
        printf("\n");
        */


        // check arrived processes
        for (int i = 0; i < n; i++) {
            if (!is_completed[i] && !is_running[i] && processes[i].arrival_time <= time) {
                is_running[i] = true;
            }
        }

        // get highest proirity of running processes
        int highest_priority = LOWEST_PRIORITY;
```

```
        int highest_priority_index = -1;
        for (int i = 0; i < n; i++) {
            if (is_running[i] && processes[i].priority < highest_priority) {
                highest_priority = processes[i].priority;
                highest_priority_index = i;
            }
        }

        // run shortest burst time process
        if (highest_priority_index != -1) {
            processes[highest_priority_index].burst_time--;
            if (processes[highest_priority_index].burst_time == 0) {
                is_running[highest_priority_index] = false;
                is_completed[highest_priority_index] = true;
                processes[highest_priority_index].turnaround_time =
processes[highest_priority_index].waiting_time + processes[highest_priority_index].const_burst_time;
            }
        }

        // calculate waiting time
        for (int i = 0; i < n; i++) {
            if (i != highest_priority_index && is_running[i]) {
                processes[i].waiting_time++;
            }
        }

        // finished all processes
        for (int i = 0; i < n; i++) {
            if (!is_completed[i]) {
                goto not_finished;
            }
        }
        break;
        not_finished:
        time++;
        continue;
```

```
    }


    // print table
    for (int i = 0; i < n; i++) {
        printf("Process id: %d, Waiting time: %d, Turnaround time: %d\n", processes[i].pid,
processes[i].waiting_time, processes[i].turnaround_time);
    }



    return 0;
}
```


task3.c - test_codespace [Codes
C Keywords | Microsoft Learn
mollahmdsaif-super-duper-succotash-xxwg5jv55qv3997g.github.dev
Google Calendar
MMS - gmail
MMS.pro - gmail
Google Classroom
BracU Library
BracU - lms
BracU USIS
test_codespace [Codespaces]
EXPLORER
task1.c U
task3.c U
TEST_CODESPACE [CODESPAC...
.vscode
README.md
task1
task1.c
task3
task3.c
OUTLINE
TIMELINE
task3.c > process
#include<stdio.h>
#include<stdbool.h>
#define MAX_BURST_TIME 1000000
#define LOWEST_PRIORITY 1000000
struct process {
int pid;
int arrival_time;
int const_burst_time;
int burst_time;
int priority;
int waiting_time;
int turnaround_time;
};
int main() {
int n;
printf("Enter number of processes: ");
scanf("%d", &n);
printf("\n");
struct process processes[n];
for (int i = 0; i < n; i++) {
printf("Enter process %d \"process_id arrival_time burst_time priority\": ", i + 1);
scanf("%d %d %d %d", &processes[i].pid, &processes[i].arrival_time, &processes[i].burs
processes[i].const_burst_time = processes[i].burst_time;
processes[i].waiting_time = 0;
processes[i].turnaround_time = 0;
}
bool is_running[n];
for (int i = 0; i < n; i++) {
is_running[i] = false;
}
bool is_completed[n];
for (int i = 0; i < n; i++) {
is_completed[i] = false;
PROBLEMS
OUTPUT
TERMINAL
PORTS
DEBUG CONSOLE
cppdbg: task3
Enter number of processes: 5
Enter process 1 "process_id arrival_time burst_time priority": 1 0 15 2
Enter process 2 "process_id arrival_time burst_time priority": 2 14 5 4
Enter process 3 "process_id arrival_time burst_time priority": 3 3 10 0
Enter process 4 "process_id arrival_time burst_time priority": 4 9 22 3
Enter process 5 "process_id arrival_time burst_time priority": 5 7 16 1
Process id: 1, Waiting time: 26, Turnaround time: 41
Process id: 2, Waiting time: 49, Turnaround time: 54
Process id: 3, Waiting time: 0, Turnaround time: 10
Process id: 4, Waiting time: 32, Turnaround time: 54
Process id: 5, Waiting time: 6, Turnaround time: 22
[1] + Done "/usr/bin/gdb" --interpreter=mi --tty=${DbgTerm} 0<"/tmp/Microsoft-MIEngine-In-vnknnu1q.1ab" 1>"/tmp/Microsoft-MIEngine-Out-ng1krvhi.ivu"
@mollahmdsaif ➜/workspaces/test_codespace (main X) $
Codespaces
main*+
Ln 9, Col 17
Spaces: 4
UTF-8
LF
C
Layout: US
Linux
ENG
3:25 AM