

22bps1059

one

```
File Edit View Terminal Tabs Help
1 #include <stdio.h>
2 #include <stdlib.h>
3 #define MAX 5
4 void display(int* arr, int* mems) {
5     printf("Current memory\n");
6     for(int i = 0; i<MAX; i++) {
7         printf("block %d:",i);
8         for(int j = 0; j<mems[i];j++) {
9             printf("%d", *arr[i]);
10            arr[i]++;
11        }printf("\n");
12    }
13    for(int j = 0; j<mems[i];j++) {
14        *arr[i]--; // resetting the pointer
15    }
16 }
17 }
18 int worstfit(int** arr, int* mems, int request) {
19     printf("Allocating %d, using worst fit\n", request);
20     int minIndex = 0;
21     for(int i = 0; i<MAX; i++) {
22         printf("minIndex is %d\n", minIndex);
23         if(mems[i]<request) continue; // cause not possible
24         printf("%d, %d, mems[%d]-request, mems[minIndex]-request;\n",
25         if(mems[i]> mems[minIndex]) {
26             minIndex = i;
27         }
28         printf("minIndex is %d", minIndex);
29     }
30 }
31 }
32 if(*arr[minIndex] == 0) { // checking if it is unallocated
33     for(int j = 0; j<request; j++) {
34         *arr[minIndex] = 1;
35         *arr[minIndex]++;
36     }
37     for(int j = 0; j<request;j++) {
38         *arr[minIndex]--; // resetting the pointer
39     }
40 }
41 return 0;
42 }
43 }
44 int bestfit(int** arr, int* mems, int request) {
45     printf("Allocating %d, using best fit\n", request);
46     int perIndex = 0;
47     for(int i = 0; i<MAX; i++) {
48         if(mems[i]==request){
49             perIndex = i;
50             break;
51         }
52     };
53 if(*arr[perIndex] == 0) { // checking if it is unallocated
54     for(int j = 0; j<request; j++) {
55         *arr[perIndex] = 1;
56         *arr[perIndex]++;
57     }
58     for(int j = 0; j<request;j++) {
59         *arr[perIndex]--; // resetting the pointer
60     }
61 }
62 }
63 if(firstfit(int** arr, int* mems, int request) {
64     printf("Allocating %d, using first fit\n", request);
65     int perIndex = 0;
66     for(int i = 0; i<MAX; i++) {
67         if(mems[i]==request){
68             perIndex = i;
69             break;
70         }
71     }
72     if(*arr[perIndex] == 0) { // checking if it is unallocated
73         for(int j = 0; j<request; j++) {
74             *arr[perIndex] = 1;
75             *arr[perIndex]++;
76         }
77     }
78     if(*arr[perIndex] == 0) { // checking if it is unallocated
79         for(int j = 0; j<request; j++) {
80             *arr[perIndex] = 1;
81             *arr[perIndex]++;
82         }
83         for(int j = 0; j<request;j++) {
84             *arr[perIndex]--;
85         }
86     }
87 }
88 }
89 int main() {
90     int *arr[MAX] = {};
91     int mems[MAX] = {};
92 }
93 // allocating memory
94 for(int i = 0; i<MAX; i++) {
95     int mem = rand()%10+1;
96     mems[i] = mem;
97     printf("Allocating %d mem\n", mem);
98     arr[i] = calloc(mem, sizeof(int));
99 }
100 // displaying memory
101 display(arr, mems);
102 display(arr, mems);
103 worstfit(arr, mems, 5);
104 display(arr, mems);
105 display(arr, mems);
106 bestfit(arr, mems, 6);
107 display(arr, mems);
108 display(arr, mems);
109 display(arr, mems);
110 firstfit(arr, mems, 4);
111 display(arr, mems);
112 display(arr, mems);
113 }
114
current memory
block 0:0000
block 1:00000000
block 2:00000000
block 3:000000
block 4:00000
allocating 5, using worst fit
minIndex is 0
minIndex is 0
2, 1minIndex is 1minIndex is 1
3, 2minIndex is 2minIndex is 2
1, 3minIndex is 2
current memory
block 0:0000
block 1:00000000
block 2:11111000
block 3:000000
block 4:00000
current memory
block 0:0000
block 1:00000000
block 2:11111000
block 3:000000
block 4:00000
allocating 6, using best fit
current memory
block 0:0000
block 1:00000000
block 2:11111000
block 3:111111
block 4:00000
current memory
block 0:0000
block 1:00000000
block 2:11111000
block 3:111111
block 4:00000
allocating 4, using first fit
current memory
block 0:1111
block 1:00000000
block 2:11111000
block 3:111111
block 4:00000
card0:/vts/os/lab10
[0] 0: bash*                                62,1      Bot
                                                "moon" 13:27 01-Nov-23
```

two

```
File Edit View Terminal Tabs Help

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <stropts.h>
4 #include <limits.h>
5 
6 #define NUM_FRAMES 3 // Number of frames in memory
7 
8 int frames[NUM_FRAMES]; // Memory frames
9 int referenceString[] = {7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2}; // Page refer
10 enum string
11 {
12     OPTIMAL,
13     FIFO
14 };
15 
16 void initializeFrames() {
17     for (int i = 0; i < NUM_FRAMES; i++) {
18         frames[i] = -1;
19     }
20 }
21 
22 void displayFrames() {
23     printf("Frames: ");
24     for (int i = 0; i < NUM_FRAMES; i++) {
25         if (frames[i] == -1) {
26             printf("(Empty)");
27         } else {
28             printf("%d ", frames[i]);
29         }
30     }
31     printf("\n");
32 }
33 
34 // Check if a page is in memory frames
35 bool isPageInFrames(int page) {
36     for (int i = 0; i < NUM_FRAMES; i++) {
37         if (frames[i] == page) {
38             return true;
39         }
40     }
41     return false;
42 }
43 
44 // FIFO Page Replacement Algorithm
45 void FIFO() {
46     initializeFrames();
47     int front = 0;
48 
49     printf("FIFO Page Replacement:\n");
50     for (int i = 0; i < n; i++) {
51         int page = referenceString[i];
52         if (!isPageInFrames(page)) {
53             pageFaults++;
54             frames[front] = page;
55             front = (front + 1) % NUM_FRAMES;
56         }
57     }
58     displayFrames();
59     printf("Page Faults: %d\n", pageFaults);
60 }
61 
62 // OPTIMAL Page Replacement Algorithm
63 void OPTIMAL() {
64     initializeFrames();
65 
66     printf("OPTIMAL Page Replacement:\n");
67     for (int i = 0; i < n; i++) {
68         int page = referenceString[i];
69         if (!isPageInFrames(page)) {
70             pageFaults++;
71             if (frames[i] != -1) {
72                 frames[i] = page;
73                 break;
74             }
75         }
76     }
77     displayFrames();
78     printf("Page Faults: %d\n", pageFaults);
79 }
80 
81 // MRU Page Replacement
82 void MRU() {
83     initializeFrames();
84 
85     printf("MRU Page Replacement:\n");
86     for (int i = 0; i < n; i++) {
87         int page = referenceString[i];
88         if (!isPageInFrames(page)) {
89             pageFaults++;
90             int farthest = -1;
91             int index = 0;
92             for (int j = 0; j < NUM_FRAMES; j++) {
93                 if (frames[j] == -1) {
94                     index = j;
95                     break;
96                 }
97                 if (frames[j] == referenceString[k]) {
98                     farthest = k;
99                     index = j;
100                }
101            }
102        }
103    }
104    if (k == n) {
105        index = 0;
106        break;
107    }
108    frames[index] = page;
109 }
110 
111 displayFrames();
112 
113 printf("Page Faults: %d\n", pageFaults);
114 }
115 
116 main() {
117     FIFO();
118     pageFaults = 0;
119     MRU();
120     pageFaults = 0;
121     OPTIMAL();
122 }

two.c 122L, 3027B written
[0] 0:bash*
```

22bps1059

first

```
File Edit View Terminal Tabs Help
1 #include <stdio.h>
2 #include <pthread.h>
3 #include <sys/types.h>
4 #include <sys/stat.h>
5 #include <sys/sem.h>
6 #include <unistd.h>
7
8 #define BUFFER_SIZE 5
9 #define NUM_PRODUCERS 2
10#define NUM_CONSUMERS 3
11
12 int semid;
13 int buffer[BUFFER_SIZE];
14 int in = 0, out = 0;
15
16 void produce(int item) {
17     buffer[in] = item;
18     in = (in + 1) % BUFFER_SIZE;
19 }
20
21 int consume() {
22     int item = buffer[out];
23     out = (out + 1) % BUFFER_SIZE;
24     return item;
25 }
26
27 void* producer(void* arg) {
28     int producer_id = *((int*)arg);
29     int item = 1;
30
31     while (1) {
32         // Produce item
33         printf("Producer %d is producing item %d.\n", producer_id, item);
34         produce(item);
35
36         // Sleep for a random time
37         sleep(1);
38
39         item++;
40     }
41 }
42
43 void* consumer(void* arg) {
44     int consumer_id = *((int*)arg);
45
46     while (1) {
47         // Consume item
48         int item = consume();
49         printf("Consumer %d is consuming item %d.\n", consumer_id, item);
50
51         // Sleep for a random time
52         sleep(2);
53     }
54 }
55
56 int main() {
57     int producer_ids[NUM_PRODUCERS];
58     int consumer_ids[NUM_CONSUMERS];
59
60     pthread_t producers[NUM_PRODUCERS];
61     pthread_t consumers[NUM_CONSUMERS];
62
63     // Create a semaphore set with two semaphores (producer and consumer)
64     semid = semget(IPC_PRIVATE, 2, IPC_CREAT | 0666);
65
66     // Initialize producer semaphore to buffer size (5)
67     semctl(semid, 0, SETVAL, BUFFER_SIZE);
68     // Initialize consumer semaphore to 0 (no items to consume initially)
69     semctl(semid, 1, SETVAL, 0);
70
71     for (int i = 0; i < NUM_PRODUCERS; i++) {
72         producer_ids[i] = i;
73         pthread_create(&producers[i], NULL, producer, &producer_ids[i]);
74
75         for (int i = 0; i < NUM_CONSUMERS; i++) {
76             consumer_ids[i] = i;
77             pthread_create(&consumers[i], NULL, consumer, &consumer_ids[i]);
78         }
79
80         for (int i = 0; i < NUM_PRODUCERS; i++) {
81             pthread_join(producers[i], NULL);
82         }
83
84         for (int i = 0; i < NUM_CONSUMERS; i++) {
85             pthread_join(consumers[i], NULL);
86         }
87
88         // Cleanup the semaphore set
89         semctl(semid, 0, IPC_RMID);
90
91     }
92     return 0;
93 }
```

cardi-vit/os/lab9 ./one
Producer 0 is producing item 1.
Producer 0 is producing item 1.
Consumer 0 is consuming item 0.
Consumer 1 is consuming item 0.
Consumer 2 is consuming item 0.
Producer 0 is producing item 2.
Producer 1 is producing item 2.
Producer 0 is producing item 3.
Producer 1 is producing item 3.
Consumer 0 is consuming item 2.
Consumer 1 is consuming item 2.
Consumer 2 is consuming item 2.
Producer 0 is producing item 3.
Producer 1 is producing item 3.
Producer 0 is producing item 4.
Producer 1 is producing item 4.
Consumer 0 is consuming item 4.
Producer 0 is producing item 5.
Producer 1 is producing item 5.
Consumer 1 is consuming item 4.
Consumer 2 is consuming item 5.
Producer 0 is producing item 6.
Producer 1 is producing item 6.
Consumer 0 is consuming item 5.
Consumer 2 is consuming item 6.
Producer 1 is consuming item 6.
Producer 0 is producing item 7.
Producer 1 is producing item 7.
Producer 0 is producing item 8.
Producer 1 is producing item 8.
Consumer 2 is consuming item 7.
Consumer 0 is consuming item 7.
Consumer 1 is consuming item 7.
Producer 0 is producing item 9.
Producer 1 is producing item 9.
Producer 0 is producing item 10.
Producer 0 is producing item 10.
Consumer 1 is consuming item 10.
Consumer 2 is consuming item 8.
Consumer 0 is consuming item 11.
Producer 1 is producing item 11.
Producer 0 is producing item 11.
Producer 0 is producing item 12.
Producer 1 is producing item 12.
Consumer 0 is consuming item 11.
Consumer 1 is consuming item 12.
Consumer 2 is consuming item 12.
Producer 0 is producing item 13.
Producer 1 is producing item 13.
^C
cardi-vit/os/lab9 scrot --focused fig1.png

[0] 0: bash* "moon" 10:33 22-Oct-24

second

```

File Edit View Terminal Tabs Help
1 #include <stdio.h>
2 #include <pthread.h>
3 #include <sys/types.h>
4 #include <sys/ipc.h>
5 #include <sys/sem.h>
6 #include <unistd.h>
7 #include <stdbool.h>
8
9 #define NUM_PHILOSOPHERS 5
10 #define EATING 0
11 #define THINKING 1
12 #define HUNGRY 2
13
14 int semid;
15 int total_eat_count = 0;
16
17 void grab_forks(int philosopher_id) {
18     struct sembuf sop[2];
19
20     sop[0].sem_num = philosopher_id;
21     sop[0].sem_op = -1;
22     sop[0].sem_flg = 0;
23
24     sop[1].sem_num = (philosopher_id + 1) % NUM_PHILOSOPHERS;
25     sop[1].sem_op = -1;
26     sop[1].sem_flg = 0;
27
28     semop(semid, sop, 2);
29 }
30
31 void put_away_forks(int philosopher_id) {
32     struct sembuf sop[2];
33
34     sop[0].sem_num = philosopher_id;
35     sop[0].sem_op = 1;
36     sop[0].sem_flg = 0;
37
38     sop[1].sem_num = (philosopher_id + 1) % NUM_PHILOSOPHERS;
39     sop[1].sem_op = 1;
40     sop[1].sem_flg = 0;
41
42     semop(semid, sop, 2);
two.c                                         1,1      Top two.c [R0]                                         84,1      Bot
:NERDTreeToggle                               :NERDTreeToggle
[0] 0:bash

```

```

65     int main() {
66         int philosopher_ids[NUM_PHILOSOPHERS];
67         pthread_t philosophers[NUM_PHILOSOPHERS];
68
69         // Create a semaphore set with one semaphore per philosopher
70         semid = semget(IPC_PRIVATE, NUM_PHILOSOPHERS, IPC_CREAT | 0666);
71
72         for (int i = 0; i < NUM_PHILOSOPHERS; i++) {
73             semctl(semid, i, SETVAL, 1); // Initialize semaphores to 1 (available)
74             philosopher_ids[i] = i;
75             pthread_create(&philosophers[i], NULL, philosopher_ids[i]);
76         }
77
78         for (int i = 0; i < NUM_PHILOSOPHERS; i++) {
79             pthread_join(philosophers[i], NULL);
80         }
81
82         // Cleanup the semaphore set
83         semctl(semid, 0, IPC_RMID);
84
85         return 0;
86     }
87 }

Philosopher 0 is eating.
Philosopher 2 is done eating and putting away forks.
Philosopher 2 is thinking.
Philosopher 3 is hungry and grabbing forks.
Philosopher 3 is eating.
Philosopher 0 is done eating and putting away forks.
Philosopher 0 is thinking.
Philosopher 1 is hungry and grabbing forks.
Philosopher 1 is eating.
Philosopher 3 is done eating and putting away forks.
Philosopher 3 is thinking.
Philosopher 1 is done eating and putting away forks.
Philosopher 1 is thinking.
Philosopher 4 is hungry and grabbing forks.
Philosopher 4 is eating.
Philosopher 2 is hungry and grabbing forks.
Philosopher 2 is eating.
Philosopher 2 is done eating and putting away forks.
Philosopher 0 is hungry and grabbing forks.
Philosopher 0 is eating.
Philosopher 3 is hungry and grabbing forks.
Philosopher 3 is eating.
Philosopher 4 is done eating and putting away forks.
Philosopher 4 is thinking.
Philosopher 2 is thinking.
Philosopher 0 is done eating and putting away forks.
Philosopher 0 is thinking.
Philosopher 3 is done eating and putting away forks.
Philosopher 3 is thinking.
Philosopher 4 is hungry and grabbing forks.
Philosopher 4 is eating.
Philosopher 1 is hungry and grabbing forks.
Philosopher 1 is eating.
^C
cardi~/vit/os/lab9 scrot --focused fig2.png
"moon" 10:31 22-Oct-23

```

third

```

File Edit View Terminal Tabs Help
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <pthread.h>
4
5 #define N 10
6
7 pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
8 pthread_cond_t empty = PTHREAD_COND_INITIALIZER;
9 pthread_cond_t full = PTHREAD_COND_INITIALIZER;
10 int count = 0;
11 char buf[N];
12
13 void monenter() {
14     pthread_mutex_lock(&mutex);
15 }
16
17 void monexit() {
18     pthread_mutex_unlock(&mutex);
19 }
20
21 void moninsert(char alpha) {
22     monenter();
23     while (count == N) {
24         printf("Buffer is full. Waiting...\n");
25         pthread_cond_wait(&full, &mutex);
26     }
27     buf[(count++) % N] = alpha; // Insert alpha into buf, wrapping around
when necessary
28     printf("Produced: %c\n", alpha);
29     if (count == 1) {
30         pthread_cond_signal(&empty);
31     }
32     monexit();
33 }
34
35 char monremove() {
36     monenter();
37     while (count == 0) {
38         printf("Buffer is empty. Waiting...\n");
39         pthread_cond_wait(&empty, &mutex);
40     }
41     char item = buf[--count % N]; // Remove an item from buf, wrapping around
:NERDTreeToggle
[0] 0:bash*                                         three.c [R0]                               2,1           Top :NERDTreeToggle                                         81,0-1      Bot :NERDTreeToggle
^C
cardi~/vit/os/lab9 scrot --focus^C
cardi~/vit/os/lab9 scrot --focused three.png
scrot: unrecognized option '--focusedthree.png'
'
cardi~/vit/os/lab9 scrot --focusedthree.png
scrot: unrecognized option '--focusedthree.png'
'
cardi~/vit/os/lab9 scrot --focused fig3.png
"
"moon" 10:30 22-Oct-23

```

fourth

fifth

22bps1059

first

second

```
File Edit View Terminal Tabs Help
Untitled
9 int main() {
10     pid_t p;
11     p = fork();
12     if (p < 0) {
13         perror("fork fail");
14         exit(1);
15     } else if (p == 0) {
16         // Child process
17         int shmid;
18         char *shared_memory;
19         shmid = shmget((key_t)2345, 1024, 0666 | IPC_CREAT);
20         if (shmid == -1) {
21             perror("shmget failed");
22             exit(1);
23         }
24         shared_memory = shmat(shmid, NULL, 0);
25         printf("Waiting for data from the parent...\n");
26         sleep(5); // Wait for the parent to write data
27         printf("Data received from the parent: %s\n", shared_memory);
28         printf("Enter data to share with the parent: ");scanf("%[^\n]s", shared_memory);
29     } else {
29         // Parent process
30         int shmid;
31         char *shared_memory;
32         shmid = shmget((key_t)2345, 1024, 0666);
33         if (shmid == -1) {
34             perror("shmget failed");
35             exit(1);
36         }
37         shared_memory = shmat(shmid, NULL, 0);
38
39         printf("Enter data to share with the child: ");scanf("%[^\n]s", shared_memory);
40         printf("Data written to shared memory: %s\n", shared_memory);
41         sleep(10);
42         printf("Data received from the child: %s\n", shared_memory);
43     }
44 }
45
46 return 0;
two.c
two.c" 49L, 1392B written
[0] 0:bash*
```

xcardi-/vit/os/lab8 sudo ./two
xEntered data to share with the child: Waiting for data from the parent...
xmoon river
xData written to shared memory: moon river
xData received from the parent: moon river
xEnter data to share with the parent: wider than
xData received from the child: wider than
xcardi-/vit/os/lab8 scrot --focused two.png

third

File Edit View Terminal Tabs Help

Untitled

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <sys/types.h>
5 #include <sys/ipc.h>
6 #include <sys/shm.h>
7 #include <sys/wait.h>
8
9 #define N 5 // Number of philosophers
10 #define THINKING 0
11 #define HUNGRY 1
12 #define EATING 2
13 #define MAX_EAT_COUNT 1 // Number of times a philosopher eats before program ends
14
15 int *state;
16 int *eat_count;
17 int shmid;
18
19 void grab_forks(int phil_id) {
20     state[phil_id] = HUNGRY;
21     printf("Philosopher %d is hungry\n", phil_id);
22     int left = (phil_id + N - 1) % N;
23     int right = (phil_id + 1) % N;
24
25     if (state[left] != EATING && state[right] != EATING) {
26         state[phil_id] = EATING;
27         printf("Philosopher %d is eating\n", phil_id);
28     }
29 }
30
31 void put_forks(int phil_id) {
32     state[phil_id] = THINKING;
33     printf("Philosopher %d is thinking\n", phil_id);
34     int left = (phil_id + N - 1) % N;
35     int right = (phil_id + 1) % N;
36
37     grab_forks(left);

```

three.c 13,23 Top three.c [+] [R0] 1 line less; before #4 1 second ago 37,20 97%

cardi~/vit/os/lab8 make three
cc three.c -o three
cardi~/vit/os/lab8 ./three
Philosopher 0 is thinking
Philosopher 1 is thinking
Philosopher 2 is thinking
Philosopher 3 is thinking
Philosopher 4 is thinking
Philosopher 0 is hungry
Philosopher 0 is eating
Philosopher 1 is hungry
Philosopher 2 is hungry
Philosopher 2 is eating
Philosopher 3 is hungry
Philosopher 4 is hungry
Philosopher 0 is thinking
Philosopher 1 is thinking
Philosopher 4 is hungry
Philosopher 0 is hungry
Philosopher 4 is eating
Philosopher 2 is hungry
Philosopher 1 is hungry
Philosopher 2 is eating
Philosopher 2 is thinking
Philosopher 1 is hungry
Philosopher 3 is hungry
Philosopher 3 is thinking
Philosopher 2 is hungry
Philosopher 2 is eating
Philosopher 4 is hungry
Philosopher 4 is eating
Philosopher 4 is thinking
Philosopher 3 is hungry
Philosopher 0 is hungry
Philosopher 0 is eating

[1] 0:bash* "moon" 11:06 12-Oct-23

22bps1059

first

```

File Edit View Search Terminal Help
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <string.h>
4 #include <stdlib.h>
5 #include <sys/types.h>
6
7 int main() {
8     int fd[2];
9     pid_t childpid;
10    char* string = "nobara";
11    char readbuffer[20]; // Define a buffer for reading data
12    ssize_t nbytes; // Use ssize_t for read/write return values
13
14    if (pipe(fd) == -1) {
15        perror("Pipe");
16        exit(1);
17    }
18
19    if ((childpid = fork()) == -1) {
20        perror("Fork");
21        exit(1);
22    }
23
24    if (childpid == 0) {
25        close(fd[0]); // Close the read end of the pipe in the child
26        write(fd[1], string, strlen(string) + 1);
27        close(fd[1]); // Close the write end after writing
28    } else {
29        close(fd[1]); // Close the write end of the pipe in the parent
30        nbytes = read(fd[0], readbuffer, sizeof(readbuffer));
31        close(fd[0]); // Close the read end after reading
32
33        if (nbytes < 0) {
34            perror("Read");
35            exit(1);
36        } else if (nbytes == 0) {
37            printf("No data received from child.\n");
38        } else {
39            printf("Received string: %s\n", readbuffer);
40        }
41    }
42
43    return 0;
44 }
one.c                                         1,1          Top

```

[5] 0:bash*

"thecuber-ThinkPad-E14" 14:36 01-Oct-23

second

```

File Edit View Search Terminal Help
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <string.h>
4 #include <stdlib.h>
5 #include <sys/types.h>
6 #include <sys/wait.h>
7 //sem_t sync;
8 int main() {
9     int ptc_pipe[2];
10    int ctp_pipe[2];
11    pid_t childpid;
12
13    char* parentToChild = "loveptc";
14    char* childToParent = "lovectp";
15    char readbufferptc[20];
16    char readbufferctp[20];
17    ssize_t ptcbytes;
18    ssize_t ctpbytes;
19
20    if(pipe(ptc_pipe) == -1 || pipe(ctp_pipe) == -1) {
21        perror("pipe");
22        exit(1);
23    }
24
25    if((childpid = fork()) == -1) {
26        perror("fork");
27    }
28
29    if(childpid == 0) {
30        printf("child pid esta %d\n", getpid());
31
32        //these will only be useful for parent
33        close(ptc_pipe[1]); // close write end of ptc pipe
34        close(ctp_pipe[0]); // close read end of ctp pipe
35
36        ptcbytes = read(ptc_pipe[0], readbufferptc, sizeof(readbufferptc));
37
38        if(ptcbytes < 0) {
39            perror("Read");
40            exit(1);
41        } else if (ptcbytes == 0) {
42            printf("No data received from parent. \n");
43        } else {
44
45            } else if (ptcbytes == 0) {
46                printf("No data received from parent. \n");
47            } else {
48                printf("Received string in child: %s\n", readbufferptc);
49            }
50        }
51    } else {
52        printf("parent pid esta %d\n", getpid());
53
54        //these will only be useful for child
55        close(ctp_pipe[1]);
56        close(ptc_pipe[0]);
57
58        write(ptc_pipe[1], parentToChild, strlen(parentToChild) +
59        1);
59        close(ptc_pipe[1]); // close write
60
61        ctpbytes = read(ctp_pipe[0], readbufferctp, sizeof(readbufferctp));
62
63        if(ctpbytes < 0) {
64            perror("Read");
65            exit(1);
66        } else if (ctpbytes == 0) {
67            printf("No data received from child. \n");
68        } else {
69            printf("Received string in parent: %s\n", readbufferctp);
70        }
71    }
72
73    wait(NULL);
74 }
75
76 return 0;
76 }
two.c                                         1,1          Top two.c [R0]

```

[5] 0:bash*

"thecuber-ThinkPad-E14" 14:38 01-Oct-23

third

File Edit View Search Terminal Help	1 // C program to implement one side of FIFO 2 // This side writes first, then reads 3 #include <stdio.h> 4 #include <string.h> 5 #include <fcntl.h> 6 #include <sys/stat.h> 7 #include <sys/types.h> 8 #include <unistd.h> 9 int main(){ 10 int fd; 11 12 // FIFO file path 13 char * myfifo = "/tmp/myfifo"; 14 15 // Creating the named file(FIFO) 16 // mkfifo(<pathname>, <permission>) 17 mkfifo(myfifo, 0666); 18 19 char arr1[80], arr2[80]; 20 while (1) 21 { 22 // Open FIFO for write only 23 fd = open(myfifo, O_WRONLY); 24 25 // Take an input arr2ing from user. 26 // 80 is maximum length 27 fgets(arr2, 80, stdin); 28 29 // Write the input arr2ing on FIFO 30 // and close it 31 write(fd, arr2, strlen(arr2)+1); 32 close(fd); 33 34 // Open FIFO for Read only 35 fd = open(myfifo, O_RDONLY); 36 37 // Read from FIFO 38 read(fd, arr1, sizeof(arr1)); 39 40 // Print the read message 41 printf("User2: %s\n", arr1); 42 close(fd); 43 44 :!scrot --focused three.png 45 [No write since last change]	1 // C program to implement one side of FIFO 2 // This side reads first, then writes 3 #include <stdio.h> 4 #include <string.h> 5 #include <fcntl.h> 6 #include <sys/stat.h> 7 #include <sys/types.h> 8 #include <unistd.h> 9 10 int main() 11 { 12 int fd1; 13 14 // FIFO file path 15 char * myfifo = "/tmp/myfifo"; 16 17 // Creating the named file(FIFO) 18 // mkfifo(<pathname>, <permission>) 19 mkfifo(myfifo, 0666); 20 21 char str1[80], str2[80]; 22 while (1) 23 { 24 // First open in read only and read 25 fd1 = open(myfifo, O_RDONLY); 26 read(fd1, str1, 80); 27 28 // Print the read string and close 29 printf("User1: %s\n", str1); 30 close(fd1); 31 32 // Now open in write mode and write 33 // string taken from user. 34 fd1 = open(myfifo, O_WRONLY); 35 fgets(str2, 80, stdin); 36 write(fd1, str2, strlen(str2)+1) 37 ; 38 } 39 return 0; threeoth.c 1,1 Top	cardi~/coding/os/27seplab ./threeoth User2: moon User1: river User1: wider User1: than User2: a User2: hello User1: my name is User2: what is it User1: its john cena User1: hello User1: my name is User1: what is it User1: its john cena
-------------------------------------	---	---	--

fourth

```
File Edit View Search Terminal Help
11 struct msg_buffer {
12     long msg_type;
13     char msg_text[MAX_MESSAGE_SIZE];
14 };
15
16 int main() {
17     key_t key;
18     int msgid;
19     struct msg_buffer message;
20
21     // Generate a unique key for the message queue
22     key = ftok("/tmp", 'A');
23     if (key == -1) {
24         perror("ftok");
25         exit(1);
26     }
27
28     // Create a message queue (the same key as the receiver)
29     msgid = msgget(key, 0666 | IPC_CREAT);
30     if (msgid == -1) {
31         perror("msgget");
32         exit(1);
33     }
34
35     while (1) {
36         // Get user input
37         printf("Enter message: ");
38         fgets(message.msg_text, sizeof(message.msg_text), stdin);
39
40         // Send the message
41         message.msg_type = 1;
42         if (msgsnd(msgid, &message, strlen(message.msg_text), 0) == -1) {
43             perror("msgsnd");
44             exit(1);
45         }
46     }
47
48     return 0;
49 }
50
four.c          26,5      Bot fouroUTH.c        4,1      60%
"four.c" 50L, 1021B written
[5] 0:bash#
```

The screenshot shows a terminal session with two panes. The left pane contains the source code for `four.c`, which creates a message queue, sends messages, and receives them. The right pane contains the source code for `fouroUTH.c`, which receives messages and prints them. The bottom pane shows the terminal command `scrot --focused four.png` being run to capture the screen.

22bps1059

first

```

File Edit View Search Terminal Help
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <unistd.h>
4 #include <pthread.h>
5
6 void userfunction() {
7     fflush(stdout);
8     printf("just a user function\n");
9 }
10 void* worker() {
11     //calls a system call sleep
12     userfunction();
13     printf("This is thread1 id: %ld\n", pthread_self());
14     sleep(1);
15     pthread_exit(0);
16 }
17
18 void* worker2() {
19     printf("This is thread2 id: %ld\n", pthread_self());
20     sleep(2);
21     pthread_exit(0);
22 }
23 int main() {
24     pthread_t t1;
25
26     int start = clock();
27     if( pthread_create(&t1, NULL, worker, NULL ) != 0 ) {
28         perror("cannot create thread\n");
29         exit(1);
30     }
31     int end = clock();
32     printf("time taken by thread 1 is %d\n", end - start);
33
34     pthread_t t2;
35     int start1 = clock();
36     if( pthread_create(&t2, NULL, worker2, NULL) !=0 ) {
37         perror("error creating thread 2");
38         exit(1);
39     }
40     int end1 = clock();
41     printf("time taken by thread 2 is %d\n", end1 - start1);
42
43     pthread_join(t1,NULL);
44     pthread_join(t2,NULL);    return 0;
}
onetime.c                                         1,1          All
[0] 0:bash*                                         "thecuber-ThinkPad-E14" 15:46 29-Sep-23

```

second

```

File Edit View Search Terminal Help
1 #include <pthread.h>
2 #include <stdio.h>
3 #include <semaphore.h>
4 #include <unistd.h>
5
6 void* f1O();
7 void* f2O();
8
9 int shared = 1;
10 sem_t s;
11
12 int main() {
13     sem_init(&s,0,1);
14     pthread_t thread1,thread2;
15     pthread_create(&thread1,NULL,f1,NULL);
16     pthread_create(&thread2,NULL,f2,NULL);
17     pthread_join(thread1,NULL);
18     pthread_join(thread2,NULL);
19     printf("Final value of shares is %d\n", shared);
20 }
21 void* f1O {
22     int x;
23     sem_wait(&s);
24     x = shared;
25     printf("Thread1 reads the value as %d\n", x);
26     x++;
27     printf("local updation by thread 1: %d\n", x);
28     sleep(1);
29     shared=x;
30     printf("Value of shared variable updated by thread 1 is %d\n", shared);
31     sem_post(&s);
32 }
33 void* f2O {
34     int y;
35     sem_wait(&s);
36     y = shared;
37     printf("Thread 2 reads the value of %d\n", y);
38     y--;
39     printf("local updation by thread2: %d\n", y);
40     sleep(1);
41     shared = y;
42     printf("value of shared variable updated by thread2 is %d\n", shared);
43     sem_post(&s);
44 }
twosemaphore.c [+]                           42,1          All
:ALEDisable
[0] 0:bash*                                         "thecuber-ThinkPad-E14" 15:48 29-Sep-23

```

third

```

File Edit View Search Terminal Help
1 #include <semaphore.h>
2 #include <unistd.h>
3 #include <stdio.h>
4 #include <pthread.h>
5 void* produceControl(void*); // producer()
6 void* consumerControl(void*); // consumer()
7 void producer(); // producer()
8 void consumer(); // consumer()
9 int current_index = 0; // current index
10 int field[10] = { }; // array
11 sem_t arraylock; // array lock
12 sem_t indexlock; // index lock
13 int main() {
14     sem_init(&arraylock, 0, 1); // init array lock
15     sem_init(&indexlock, 0, 1); // init index lock
16     pthread_t consumer_thread, producer_thread; // threads
17     printf("initial array\n");
18     for(int i = 0; i<10; i++) printf("%d ", field[i]); // print initial array
19     pthread_create(&producer_thread, NULL, produceControl, NULL); // create producer thread
20     pthread_create(&consumer_thread, NULL, consumerControl, NULL); // create consumer thread
21     pthread_join(&producer_thread, NULL); // join producer thread
22     pthread_join(&consumer_thread, NULL); // join consumer thread
23     printf("final array values \n");
24     for(int i = 0; i<10; i++) printf("%d ", field[i]); // print final array
25 }
-- 1 /* W: control reaches end of non-void function [-Wreturn-type]
26
27 void* produceControl(void* {
28     producer();
29     producer();
30     producer();
31     producer();
32 } /* W: control reaches end of non-void function [-Wreturn-type]
33
34 void producer() {
35     sem_wait(&arraylock);
36     sem_wait(&indexlock);
37     field[current_index] = 1; // set current index to 1
38     printf("item produced\n");
39     for(int i = 0; i<10; i++) printf("%d ", field[i]); // print array again
40     sem_post(&arraylock);
41     sem_post(&indexlock);
42 }
43
44 void* consumerControl(void* {
45     consumer();
46     consumer();
47 } /* W: control reaches end of non-void function [-Wreturn-type]
48
49 void consumer() {
50     sem_wait(&arraylock);
51     sem_wait(&indexlock);
52     field[--current_index] = 0; // set current index to 0
53     printf("item consumed nom nom\n");
54     for(int i = 0; i<10; i++) printf("%d ", field[i]); // print array again
55     sem_post(&arraylock);
56     sem_post(&indexlock);
57 }
58
59 threeproducer.c [R0] 57,1 Bot
cardi~/coding/os/lab20sep_lab6 make threeproducer
cc threeproducer.c -o threeproducer
cardi~/coding/os/lab20sep_lab6 ./threeproducer
initial array
0 0 0 0 0 0 0 0 0 0
item produced
1 0 0 0 0 0 0 0 0 0
item produced
1 1 0 0 0 0 0 0 0 0
item produced
1 1 1 0 0 0 0 0 0 0
item consumed nom nom
1 1 0 0 0 0 0 0 0 0
item consumed nom nom
1 1 0 0 0 0 0 0 0 0
final array values
1 1 0 0 0 0 0 0 0 0
cardi~/coding/os/lab20sep_lab6 scrot --focused three.png
[0] 0: bash* 2,1 Top
[0] 0: bash* 57,1 Bot
"thecuber-ThinkPad-E14" 15:51 29-Sep-23

```

fourth

```

File Edit View Search Terminal Help
1 #include <stdio.h>
2 #include <semaphore.h>
3 #include <unistd.h>
4 #include <pthread.h>
5 #define size 10
6
7 struct roundtable {
8     int places[size];
9 };
10
11 void printCircle(struct roundtable r1);
12 void algorithm(int left, int center,int right);
13 void* p0f(void*); // philosopher 0
14 void* p1f(void*); // philosopher 1
15 void* p2f(void*); // philosopher 2
16 void* p3f(void*); // philosopher 3
17 void* p4f(void*); // philosopher 4
18
19 sem_t forkaccess; // fork access
20 struct roundtable table; // roundtable
21 //just for printing
22
23 void* p0f(void*) {
24     sem_wait(&forkaccess); // wait for fork access
25     algorithm(0,0,1); // algorithm
26     sem_post(&forkaccess); // release fork access
27 }
28 void* p1f(void*) {
29     sem_wait(&forkaccess); // wait for fork access
30     algorithm(1,2,3); // algorithm
31     sem_post(&forkaccess); // release fork access
32 }
33 void* p2f(void*) {
34     sem_wait(&forkaccess); // wait for fork access
35     algorithm(3,4,5); // algorithm
36     sem_post(&forkaccess); // release fork access
37 }
38 void* p3f(void*) {
39     sem_wait(&forkaccess); // wait for fork access
40     algorithm(5,6,7); // algorithm
41     sem_post(&forkaccess); // release fork access
42 }
43 void* p4f(void*) {
44     sem_wait(&forkaccess); // wait for fork access
45     algorithm(7,8,9); // algorithm
46     sem_post(&forkaccess); // release fork access
47 }
48
49 int main() {
50     // here 0s represent diners and 1 represents the forks
51     // in order for a philo to be done dining they require 2 forks
52     pthread_t p1,p2,p3,p4,p0;
53     for(int i = 0; i<size; i++) {
54         if(i%2 == 0) table.places[i] = 0; // even place
55         else table.places[i] = 1; // odd place
56     }
57     printCircle(table); // print circle
58     sem_init(&forkaccess,0,1); // init fork access
59     pthread_create(&p0, NULL, p0f, NULL); // create philosopher 0
60     pthread_create(&p1, NULL, p1f, NULL); // create philosopher 1
61     pthread_create(&p2, NULL, p2f, NULL); // create philosopher 2
62     pthread_create(&p3, NULL, p3f, NULL); // create philosopher 3
63     pthread_create(&p4, NULL, p4f, NULL); // create philosopher 4
64
65     pthread_join(p0, NULL); // join philosopher 0
66     pthread_join(p1, NULL); // join philosopher 1
67     pthread_join(p2, NULL); // join philosopher 2
68     pthread_join(p3, NULL); // join philosopher 3
69     pthread_join(p4, NULL); // join philosopher 4
70
71     printf("all philosophers done eating\n");
72     printCircle(table); // print circle again
73     return 0;
74 }
75
76 void printCircle(struct roundtable r1) {
77     printf("p0.%d p1.%d\n"
78             "p2.%d p3.%d\n"
79             "p4.%d p5.%d\n"
80             "p6.%d p7.%d\n"
81             "p8.%d p9.%d\n"
82             "p10.%d\n", r1.places[0], r1.places[1],
83             r1.places[2], r1.places[3],
84             r1.places[4], r1.places[5],
85             r1.places[6], r1.places[7],
86             r1.places[8], r1.places[9],
87             r1.places[10]);
88
89
90 }
91
92 void algorithm(int left, int center, int right) {
93     printf("\n philosopher %d eat ing\n", center/2);
94     table.places[left]--; // decrease place at left
95     table.places[right]--; // decrease place at right
96     table.places[center]++; // increase place at center
97     printCircle(table); // print circle again
98     table.places[center]++; // increase place at center
99     table.places[left]++; // increase place at left
100    table.places[right]++; // increase place at right
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2049
2050
2051
2052
2053
2054
2055
205
```

22bps1059

first – sigint

```

File Edit View Search Terminal Help
1 #include <stdio.h>
2 #include <signal.h>
3 #include <sys/wait.h>
4 #include <unistd.h>
5 #include <stdlib.h>
6 #define SIGINT 2
7
8 void siginthandler() {
9     printf("handled\n");
10    exit(0);
11 }
12
13 int main() {
14     signal(SIGINT,siginthandler);
15     while(1) {
16         printf("Hello world\n");
17         sleep(1);
18     }
19     return 0;
20 }

sigint2.c          1,1      All
[0] 0:bash*          "thecuber-ThinkPad-E14" 21:49 05-Sep-23

```

second – sigint, sighup, sigquit

```

File Edit View Search Terminal Help
3 #include <signal.h>
4 #include <stdlib.h>
5 #include <unistd.h>
6 #include <sys/wait.h>
7 void sighup(); void sigint(); void sigquit();
8
9 void main() {
10     int pid;
11     pid = fork();
12     if(pid == 0) {
13         signal(SIGHUP,sighup);
14         signal(SIGQUIT,sigquit);
15         signal(SIGINT,sigint);
16         signal(SIGQUIT,sigquit);
17
18         for(;;);
19     }
20     else {
21         printf("\n parent: sending sighup\n\n");
22         kill(pid,SIGHUP);
23         sleep(3);
24         printf("\n parent: sending sigint\n\n");
25         kill(pid,SIGINT);
26         sleep(3);
27         printf("\n parent: sending sigquit\n\n");
28         kill(pid,SIGQUIT);
29         sleep(3);
30     }
31 }
32
33 void sighup() {
34     printf("CHILD: i have received a SIGHUP\n");
35     fflush(stdout);
36 }
37
38 void sigint() {
39     printf("CHILD: i have received a sigint\n");
40     fflush(stdout);
41 }
42
43 void sigquit() {
44     printf("CHILD: i have received a sigquit\n");
45     fflush(stdout);
46 }
sigint.c          28,13      Bot
"sigint.c" 46L, 989B written
[0] 0:bash*          "thecuber-ThinkPad-E14" 21:45 05-Sep-23

```

third – roundrobin

```

File Edit View Search Terminal Help
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <stdlib.h>
4 #include <signal.h>
5 #include <sys/types.h>
6 #include <sys/wait.h>
7 #define MAX_PROCESSES 5
8 #define TIME_QUANTUM 2
9 int completed = 0;
10 void execute(int pid, int time_slice) {
11     printf("Executing Process %d for %d seconds\n", pid, time_slice);
12     sleep(time_slice);
13 }
14 int main() {
15     int n = 5; // Number of processes
16     int arrival_time[] = {0, 1, 2, 3, 4}; /* W: unused variable 'arrival_time' [-Wunused-variable] */
17     int burst_time[] = {6, 2, 8, 3, 5};
18     int remaining_time[MAX_PROCESSES];
19     int child_pids[MAX_PROCESSES];
20
21     for (int i = 0; i < n; i++) {
22         remaining_time[i] = burst_time[i];
23     }
24
25     int current_process = 0; /* W: unused variable 'current_process' [-Wunused-variable] */
26     for (int i = 0; i < n; i++) {
27         pid_t child_pid = fork();
28
29         if (child_pid == -1) {
30             perror("fork");
31             exit(1);
32         }
33         else if (child_pid == 0) {
34             while (remaining_time[i] > 0) {
35                 int time_slice = (remaining_time[i] < TIME_QUANTUM) ? remaining_time[i] : TIME_QUANTUM;
36                 execute(i + 1, time_slice);
37                 remaining_time[i] -= time_slice;
38
39                 if (remaining_time[i] > 0) {
40                     printf("Process %d preempted. Remaining time: %d seconds\n", i + 1, r
41     emaining_time[i]);
42             }
43         }
44     }
45     if (completed == n) {
46         exit(0);
47     }
48     else {
49         child_pids[i] = child_pid;
50     }
51 }
52 for (int i = 0; i < n; i++) {
53     waitpid(child_pids[i], NULL, 0);
54     printf("Parent Process: Child Process %d completed.\n", child_pids[i]);
55 }
56
roundrobin.c [R0]                                     45,0-1   93%
22BPS1059>./roundrobin
Executing Process 1 for 2 seconds
Executing Process 2 for 2 seconds
Executing Process 3 for 2 seconds
Executing Process 4 for 2 seconds
Executing Process 5 for 2 seconds
Process 4 preempted. Remaining time: 1 seconds
Process 1 preempted. Remaining time: 4 seconds
Process 3 preempted. Remaining time: 6 seconds
Process 5 preempted. Remaining time: 3 seconds
Executing Process 4 for 1 seconds
Executing Process 1 for 2 seconds
Executing Process 3 for 2 seconds
Executing Process 5 for 2 seconds
Process 3 preempted. Remaining time: 4 seconds
Process 1 preempted. Remaining time: 2 seconds
Process 5 preempted. Remaining time: 1 seconds
Executing Process 3 for 2 seconds
Executing Process 5 for 1 seconds
Executing Process 1 for 2 seconds
Process 3 preempted. Remaining time: 2 seconds
Executing Process 3 for 2 seconds
Parent Process: Child Process 336688 completed.
Parent Process: Child Process 336689 completed.
Parent Process: Child Process 336690 completed.
Parent Process: Child Process 336691 completed.
Parent Process: Child Process 336692 completed.
22BPS1059>
"roundrobin.c" 57L, 1552B written
[0] 0:bash*                                         "thecuber-ThinkPad-E14" 21:37 05-Sep-23

```

fourth – srjf

```

File Edit View Search Terminal Help
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <stdlib.h>
4 #include <signal.h>
5 #include <sys/types.h>
6 #include <sys/wait.h>
7 #define MAX_PROCESSES 5
8 int completed = 0;
9
10 void execute(int pid, int time_slice) {
11     printf("Executing Process %d for %d seconds\n", pid, time_slice);
12     sleep(time_slice);
13 }
14 int main() {
15     int n = 5;
16     int arrival_time[] = {0, 1, 2, 3, 4};
17     int burst_time[] = {6, 2, 8, 3, 5};
18     int remaining_time[MAX_PROCESSES];
19     int child_pids[MAX_PROCESSES];
20     for (int i = 0; i < n; i++) remaining_time[i] = burst_time[i];
21     int current_time = 0;
22
23     for (int i = 0; i < n; i++) {
24         pid_t child_pid = fork();
25
26         if (child_pid == -1) {
27             perror("fork");
28             exit(1);
29         }
30         else if (child_pid == 0) { // Child process
31             while (remaining_time[i] > 0 && !completed) {
32                 int time_slice = 1;
33                 if (remaining_time[i] < 1) {
34                     time_slice = remaining_time[i];
35                 }
36                 execute(i + 1, time_slice);
37                 remaining_time[i] -= time_slice;
38             }
39             exit(0);
40         }
41         else { // Parent process
42             child_pids[i] = child_pid;
43         }
44     }
45     if (completed == n) {
46         exit(0);
47     }
48     else {
49         int status;
50         waitpid(child_pids[i], &status, 0);
51         if (WIFSIGNALED(status) && WTERMSIG(status) == SIGHUP) {
52             printf("Parent Process: Child Process %d terminated.\n", child_pids[i]);
53         }
54     }
55 }
56
srjf.c [R0]                                         57,1   97%
22BPS1059>./srjf
Executing Process 1 for 1 seconds
Executing Process 2 for 1 seconds
Executing Process 3 for 1 seconds
Executing Process 4 for 1 seconds
Executing Process 5 for 1 seconds
Executing Process 1 for 1 seconds
Executing Process 2 for 1 seconds
Executing Process 3 for 1 seconds
Executing Process 4 for 1 seconds
Executing Process 5 for 1 seconds
Parent Process: Child Process 336938 terminated.
Parent Process: Child Process 336939 terminated.
Parent Process: Child Process 336940 terminated.
Parent Process: Child Process 336941 terminated.
Parent Process: Child Process 336942 terminated.
22BPS1059>
"srjf.c" 58L, 1587B written
[0] 0:bash*                                         "thecuber-ThinkPad-E14" 21:39 05-Sep-23

```

22bps1059

first

```

File Edit View Search Terminal Help
1 #include <sys/wait.h>
2 #include <stdlib.h>
3 #include <stdio.h>
4
5 void pr_exit(int status) {
6     if(WIFEXITED(status)) printf("normal termination : %d\n", WEXITSTATUS(status));
7     else if(WIFSIGNALED(status)) printf("abnormal termination: %d\n", WTERMSIG(status));
8     else if (WIFSTOPPED(status)) printf("child stopped %d\n", WSTOPSIG(status));
9 }
10 pid_t fork();
11 int main() {
12     pid_t pid;
13     int status;
14
15     // process 1 (just exits)
16     if ( ( pid=fork() ) < 0 )
17         printf("fork error");
18     else if(pid == 0)
19         exit(0);
20
21     if( wait(&status) != pid )
22         printf("wait error");
23
24     pr_exit(status);
25
26     // process 2 (aborts)
27     if( (pid=fork() ) < 0 )
28         printf("fork error");
29     else if (pid == 0)
30         abort();
31
32     if( wait(&status) != pid ) printf("wait error");
33
34     pr_exit(status);
35
36
37     exit(0);
38 }

one_processExit.c          1,1      All
[0] 0:bash*               "thecuber-ThinkPad-E14" 12:30 29-Aug-23

```

second

```

File Edit View Search Terminal Help
1 #include <sys/wait.h>
2 #include <stdlib.h>
3 #include <stdio.h>
4
5 void pr_exit(int status) {
6     if(WIFEXITED(status)) printf("normal termination : %d\n", WEXITSTATUS(status));
7     else if(WIFSIGNALED(status)) printf("abnormal termination: %d\n", WTERMSIG(status));
8     else if (WIFSTOPPED(status)) printf("child stopped %d\n", WSTOPSIG(status));
9 }
10 pid_t fork();
11 int main() {
12     pid_t pid;
13     int status;
14
15     if ( ( pid=fork() ) < 0 )
16         printf("fork error");
17     else if(pid == 0)
18         printf("Waiting process...\n");
19     wait(0);
20     exit(0);
21
22     if( wait(&status) != pid )
23         printf("wait error");
24
25     pr_exit(status);
26     exit(0);
27 }

two_processwait.c          18,36    All
"two_processwait.c" 27L, 674B written
[0] 0:bash*               "thecuber-ThinkPad-E14" 12:33 29-Aug-23

```

third

```
File Edit View Search Terminal Help
1 #include <sys/wait.h>
2 #include <stdlib.h>
3 #include <stdio.h>
4 void pr_exit(int status) {
5     if(WIFEXITED(status)) printf("normal termination : %d\n", WEXITSTATUS(status));
6     else if(WIFSIGNALED(status)) printf("abnormal termination: %d\n", WTERMSIG(status));
7     else if (WIFSTOPPED(status)) printf("child stopped %d\n", WSTOPSIG(status));
8 }
9 pid_t fork(); int execvp(); int sleep(int);
10 int main() {
11     pid_t pid;int status;
12     // process 1
13     if ((pid = fork()) < 0) printf("fork error");
14     else if(pid == 0){
15         char *command = "cat";
16         char args[] = {command,"file.txt", NULL};
17         execvp(command, args);
18         exit(0);
19     }
20     if(wait(&status) != pid) printf("wait error");
21     pr_exit(status);
22     // process 2
23     if((pid = fork()) < 0) printf("fork error");
24     else if (pid == 0) {
25         abort();
26     }
27     if(wait(&status) != pid) printf("wait error");
28     pr_exit(status);
29     // process 3
30     if((pid = fork()) < 0) printf("fork error");
31     else if (pid == 0){
32         exit(0);
33     }
34     if(wait(&status) != pid) printf("wait error");
35     pr_exit(status);
36     // process 4
37     if((pid = fork()) < 0) printf("fork error");
38     else if (pid == 0){
39         sleep(2);
40         exit(0);
41     }
42     if(wait(&status) != pid) printf("wait error");
43     pr_exit(status);
44     exit(0);
}
three_entirelifecycle.c 22,5 Top
"three_entirelifecycle.c" 46L, 1318B written
[0] 0:bash*                                         "thecuber-ThinkPad-E14" 12:35 29-Aug-23
```

fourth

```
File Edit View Search Terminal Help
1 #include <sys/wait.h>
2 #include <signal.h>
3 #include <stdlib.h>
4 #include <stdio.h>
5
6 void pr_exit(int status) {
7     if(WIFEXITED(status)) printf("normal termination : %d\n", WEXITSTATUS(status));
8     else if(WIFSIGNALED(status)) printf("abnormal termination: %d\n", WTERMSIG(status));
9     else if (WIFSTOPPED(status)) printf("child stopped %d\n", WSTOPSIG(status));
10 }
11
12 int getpid(); int getppid(); int fork(); int sleep(); int pause(); int kill();
13
14 int main() {
15     int processTimes[] = {7,1,2,1,5};
16     int numofprocesses = sizeof(processTimes) / sizeof(processTimes[0]);
17     pid_t processpids[numofprocesses]; /* W: variable 'processpids' set but not used [-Wunused-parameter] */
18
19     //initializing processes
20     for(int i = 0; i<numofprocesses; i++) {
21         pid_t pid = fork();
22         int status;
23         processpids[i] = pid;
24
25         if (pid < 0)
26             perror("fork error");
27
28         else if(pid == 0){
29             printf("\nim process %i my pid is %d and my parent is %d\n",i,getpid(),getppid());
30             fflush(stdout);
31             sleep(processTimes[i]);
32             exit(0);
33         }
34
35         if( wait(&status) != pid )
36             printf("wait error\n");
37
38         pr_exit(status);
39     }
40
41     exit(0);
42     return 0;
43 }
fcfs.c 11,0-1 All
[0] 0:bash*                                         "thecuber-ThinkPad-E14" 12:36 29-Aug-23
```

fifth

```
1 //22bps1059
2 #include <stdio.h>
3 #include <stdlib.h>
4 #include <sys/types.h>
5 #include <sys/wait.h>
6 #include <unistd.h>
7
8 void swap( int *x, int *y){
9     int c = *x;
10    *x = *y;
11    *y = c;
12 }
13 int main() {
14     // array of arrival time, bursttime, processnumber
15     int processtable[5][3] = { {8,4,0},{4,10,1},{3,5,2},{31,1,3},{12,3,4} };
16     int numprocesses = 5;
17
18     // sorting accoring to bursttime to get sjfirst
19     for(int i = 0; i< numprocesses; i++) {
20         for(int j = 0; j<numprocesses-1; j++) {
21             if(processtable[j][1] > processtable[j+1][1]) {
22                 swap( &processtable[j][0],&processtable[j+1][0] );
23                 swap( &processtable[j][1],&processtable[j+1][1] );
24                 swap( &processtable[j][2],&processtable[j+1][2] );
25             }
26         }
27     }
28
29     printf("new order of processes (see the third number)\n");
30     for(int i = 0; i< numprocesses; i++) {
31         for(int j = 0; j<3; j++) {
32             printf("%d ",processtable[i][j]);
33         }printf("\n");
34     }
35 }
```

```

34     }
35
36     for(int i = 0; i<numprocesses; i++) {
37         int processdecider = processtable[i][2];
38         pid_t pid = fork(); int status;
39         if (pid == -1) {
40             perror("fork error");
41             exit(1);
42         } else if (pid == 0) {
43             sleep(processtable[i][1]);
44             switch(processdecider){
45                 case 0:
46                     fflush(stdout);
47                     printf("im the first process\n");
48                     break;
49
50                 case 1:
51                     fflush(stdout);
52                     printf("im the second process\n");
53                     break;
54
55                 case 2:
56                     fflush(stdout);
57                     printf("im the Third\n");
58                     break;
59
60                 case 3:
61                     fflush(stdout);
62                     printf("im the fourth\n");
63                     break;
64
65                 case 4:
66                     printf("im the fifth\n");
67                     break;
68             }
69             exit(0);
70         }
71         if( wait(&status) != pid )
72             printf("wait error\n");
73     }
74
75     // Wait for child processes using
76     while(wait(NULL) > 0);
77
78     printf("All child processes have finished in shortest job order.\n");
79     return 0;
80 }
81
82

```

```
File Edit View Search Terminal Help
22bps1059> ./sjf
new order of processes (see the third number)
3 1 3
12 3 4
8 4 0
3 5 2
4 10 1
im the fourth
im the fifth
im the first process
im the Third
im the second process
All child processes have finished in shortest job order.
22bps1059> █
[0] 0:bash*                                     "thecuber-ThinkPad-E14" 12:40 29-Aug-23
```

22bps1059

lab - 3

first

```

1 #include <stdio.h>
2
3 int main() {
4     fork(); /* W: implicit declaration of function 'fork' [-Wimplicit-declaration]
5     printf("%d\n", getpid()); /* W: implicit declaration of function 'getpid' [-Wimplicit-declaration]
6 }
```

~/coding/os/22bps1059 > ./one
116062
116064
~/coding/os/22bps1059 > |

second

```

1 #include <stdio.h>
2
3 int main() {
4     int a = 14; int b = 3;
5     int p1 = fork(); /* W: implicit declaration of function 'fork' [-Wimplicit-declaration]
6     int p2 = fork();
7     if(!p1 && p2) {
8         printf("this is %d, child of %d\n", getpid(), getpid());
9         printf("sum of %d and %d is %d\n", a,b,a+b);
10    }
11    else if (p1 && !p2) {
12        printf("this is %d, child of %d\n", getpid(), getpid());
13        printf("product of %d and %d is %d\n", a,b,a*b);
14    }
15    printf("\n");
16 }
```

~/coding/os/22bps1059 > ./one
this is 198009, child of 198006
product of 14 and 3 is 42

this is 198008, child of 198006
sum of 14 and 3 is 17
~/coding/os/22bps1059 > |

third

```

1 #include <stdio.h>
2
3 int main() {
4     char *command = "cat";
5     char *args[] = {command,"file.txt",NULL};
6     int pid = fork(); /* W: implicit declaration of function 'fork' [-Wimplicit-declaration]
7     if(pid>0)
8         execvp(command, args); /* W: implicit declaration of function 'execvp' [-Wimplicit-declaration]
9     return 0;
10 }
```

~/coding/os/22bps1059 > ./two
This is a file
with several lines
of text in it
~/coding/os/22bps1059 > |

fourth

```

1 #include <stdio.h>
2 int main() {
3     int p1 = fork(); /* W: implicit declaration of function 'fork' [-Wimplicit-declaration]
4     int p2 = fork();
5     if(!p1 && p2) {
6         printf("Writing to filename.txt\n");
7         char *command = "./threebin";
8         char *args[] = {command,NULL};
9         execvp(command, args); /* W: implicit declaration of function 'execvp' [-Wimplicit-declaration]
10    }
11    else if (p1 && !p2) {
12        char *command = "cat";
13        char *args[] = {command,"urregno.txt",NULL};
14        execvp(command, args);
15    }
16 } 
```

1 #include <stdio.h>
2 int main() {
3 FILE *file = fopen("urregno.txt","w");
4 if(file==NULL) {
5 printf("failed to open file\n");
6 return 1;
7 }
8 fprintf(file,"22BPS1059!\n");
9 fclose(file);
10 printf("written to file \n");
11 return 0;
12 }

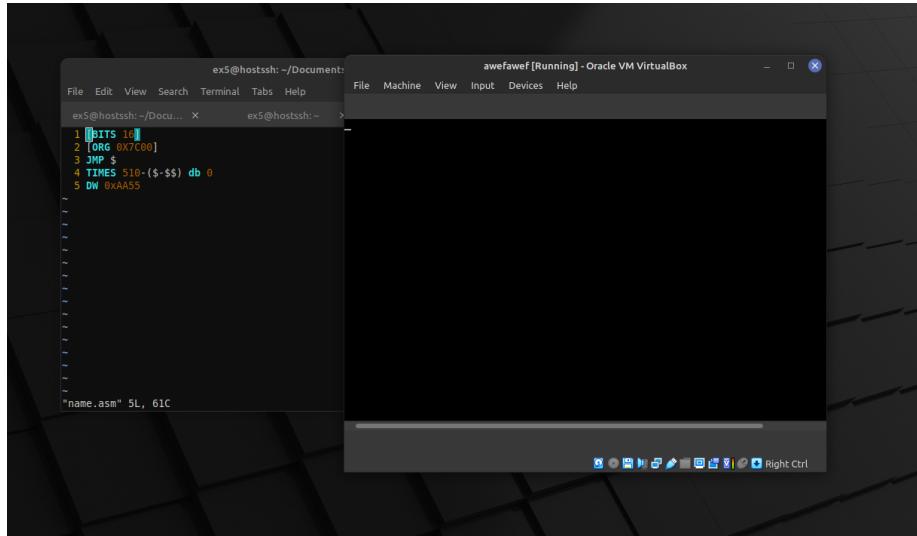
threebin.c

~/coding/os/22bps1059 > ./three
Writing to filename.txt
written to file
22BPS1059!

22bps1059

lab - 2

first

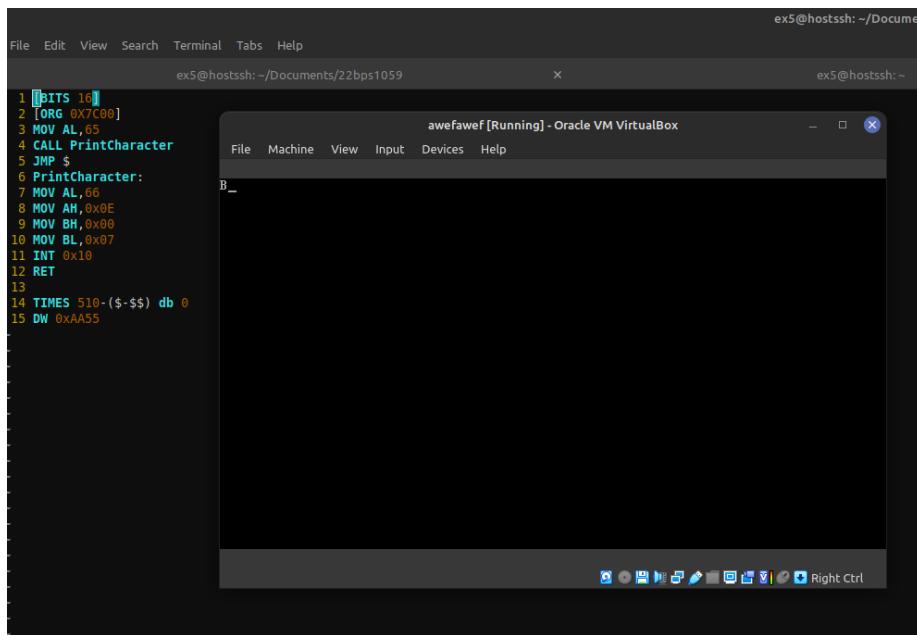


A screenshot of a terminal window titled "awefawef [Running] - Oracle VM VirtualBox". The window has two tabs: "ex5@hostssh: ~/Documents" and "ex5@hostssh: ~". The left tab contains assembly code:

```
1 [BITS 16]
2 [ORG 0X7C00]
3 JMP $
4 TIMES 510-( $$) db 0
5 DW 0xAA55
```

The right tab is empty. The desktop background is dark with a grid pattern.

second



A screenshot of a terminal window titled "awefawef [Running] - Oracle VM VirtualBox". The window has two tabs: "ex5@hostssh: ~/Documents/22bps1059" and "ex5@hostssh: ~". The left tab contains assembly code:

```
1 [BITS 16]
2 [ORG 0X7C00]
3 MOV AL,05
4 CALL PrintCharacter
5 JMP $
6 PrintCharacter:
7 MOV AL,66
8 MOV AH,0x0F
9 MOV BH,0x00
10 MOV BL,0x07
11 INT 0x10
12 RET
13
14 TIMES 510-( $$) db 0
15 DW 0xAA55
```

The right tab is empty. The desktop background is dark with a grid pattern.

third

The terminal window on the left displays assembly code for printing 'Hello World'. The virtual machine window on the right shows the output 'Hello World'.

```
1 [BITS 16]
2 [ORG 0x1C00]
3 MOV SI,HelloString
4 CALL PrintString
5 JMP $ 
6 PrintCharacter:
7 MOV AH,0xE
8 MOV BH,0x00
9 MOV BL,0x07
10 INT 0x10
11 RET
12 PrintString:
13 next_character:
14 character MOV AL,[SI]
15 INC SI
16 OR AL,AL
17 JZ exit_function
18 CALL PrintCharacter
19 JMP next_character
20 exit_function:
21 RET
22 HelloString db 'Hello World',0
23 TIMES 510-( $$) db 0
24 DW 0xAA55
```

fourth

The terminal window on the left displays assembly code for printing 'Sahil Upasane 22BPS1059'. The virtual machine window on the right shows the output 'Sahil Upasane 22BPS1059'.

```
1 [BITS 16]
2 [ORG 0x1C00]
3 MOV SI,HelloString
4 CALL PrintString
5 JMP $ 
6 PrintCharacter:
7 MOV AH,0xE
8 MOV BH,0x00
9 MOV BL,0x07
10 INT 0x10
11 RET
12 PrintString:
13 next_character:
14 character MOV AL,[SI]
15 INC SI
16 OR AL,AL
17 JZ exit_function
18 CALL PrintCharacter
19 JMP next_character
20 exit_function:
21 RET
22 HelloString db 'Sahil Upasane 22BPS1059',0
23 TIMES 510-( $$) db 0
24 DW 0xAA55
```

22bps1059

first

```
cardi~/Documents/vit/os/22bps1059 date
Wednesday 26 July 2023 07:44:16 PM IST
cardi~/Documents/vit/os/22bps1059 name="sahil"
cardi~/Documents/vit/os/22bps1059 echo $name
sahil
cardi~/Documents/vit/os/22bps1059 echo $$
410003
cardi~/Documents/vit/os/22bps1059 date
Wednesday 26 July 2023 07:44:16 PM IST
cardi~/Documents/vit/os/22bps1059 name="sahil"
cardi~/Documents/vit/os/22bps1059 echo $name
sahil
cardi~/Documents/vit/os/22bps1059 echo $$
410003
cardi~/Documents/vit/os/22bps1059 ls
cardi~/Documents/vit/os/22bps1059 cd ..
cardi~/Documents/vit/os ls
22bps1059 boat.png ostemplate.tex texput.log
cardi~/Documents/vit/os ls -l
total 84
drwxrwxr-x 2 thecuber thecuber 4096 Jul 26 19:00 22bps1059
-rw-rw-r-- 1 thecuber thecuber 72254 Mar 8 2021 boat.png
-rw-rw-r-- 1 thecuber thecuber 1877 Jul 26 18:56 ostemplate.tex
-rw-rw-r-- 1 thecuber thecuber 675 Jul 26 14:09 texput.log
cardi~/Documents/vit/os ls -a
. 22bps1059 boat.png ostemplate.tex texput.log
cardi~/Documents/vit/os cat ostemplate.tex
% This is a small sample LaTeX input file (Version of 10 April 1994)
~
1 this is text in the file
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
trial.moon [+]
1,25 All
-- INSERT --
cardi~/Documents/vit/os cd 22bps1059/
cardi~/Documents/vit/os/22bps1059 vi trial.moon
cardi~/Documents/vit/os/22bps1059 cat trial.moon
this is text in the file
cardi~/Documents/vit/os/22bps1059 cp trial.moon trialcopy.moon
cardi~/Documents/vit/os/22bps1059 ls
trialcopy.moon trial.moon
cardi~/Documents/vit/os/22bps1059 rm trial.moon
cardi~/Documents/vit/os/22bps1059 mkdir asfd
cardi~/Documents/vit/os/22bps1059 ls
asfd trialcopy.moon
cardi~/Documents/vit/os/22bps1059 rmdir asfd
rmdir: failed to remove 'asdf': No such file or directory
cardi~/Documents/vit/os/22bps1059 ls
asfd trialcopy.moon
cardi~/Documents/vit/os/22bps1059 rmdir asfd
cardi~/Documents/vit/os/22bps1059 |
cardi~/Documents/vit/os/22bps1059 chmod 777 trialcopy.moon
cardi~/Documents/vit/os/22bps1059 ps
    PID TTY      TIME CMD
410003 pts/3    00:00:00 bash
410373 pts/3    00:00:00 ps
cardi~/Documents/vit/os/22bps1059 ps -f
UID      PID  PPID C STIME TTY      TIME CMD
thecuber 410003 400032 0 19:40 pts/3    00:00:00 bash
thecuber 410374 410003 0 19:53 pts/3    00:00:00 ps -f
cardi~/Documents/vit/os/22bps1059 |
```

```

top - 19:53:36 up 2 days, 15 min, 1 user, load average: 1.56, 1.66, 1.73
Tasks: 297 total, 2 running, 295 sleeping, 0 stopped, 0 zombie
%CPU(s): 12.0 us, 4.2 sy, 0.0 ni, 83.8 id, 0.0 wa, 0.0 hi, 0.0 sl, 0.0 st
Mem Mem : 7616.0 total, 694.7 free, 3882.8 used, 3038.5 buff/cache
Mem Swap: 0.0 total, 0.0 free, 0.0 used. 2626.6 avail Mem

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
11754 thecuber 20 0 131924 715256 155164 S 41.2 9.2 197:29 29 firebox
262534 thecuber 20 0 525816 106876 R 35.3 6.7 83:34.03 Isolated Web Co
3160 thecuber 20 0 6767464 308976 84784 S 11.8 4.0 63:50.95 gnome-shell
63764 thecuber 20 0 546648 64276 35940 S 11.8 0.8 24:21.57 RRD Process
410391 thecuber 20 0 21892 4284 3424 R 5.9 0.1 0:00.03 top
1 root 20 0 167808 9984 5924 S 0.0 0.1 2:45.09 systemd
2 root 20 0 0 0 0 S 0.0 0.0 0:00.11 kthreadd
3 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 rCU_gp
4 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 rCU_par_gp
5 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 stub_flushwq
6 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 netns
10 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 mm_percpu_wq
11 root 20 0 0 0 0 I 0.0 0.0 0:00.00 rCU_tasks_kthread
12 root 20 0 0 0 0 I 0.0 0.0 0:00.00 rCU_tasks_rude_kthread
13 root 20 0 0 0 0 I 0.0 0.0 0:00.00 rCU_tasks_trace_kthread
14 root 20 0 0 0 0 S 0.0 0.0 0:02.69 ksoftirqd/0
15 root 20 0 0 0 0 I 0.0 0.0 1:05.41 rCU_preempt
16 root rt 0 0 0 0 S 0.0 0.0 0:00.42 migration/0
17 root -51 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/0
19 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/0
20 root 20 0 0 0 0 S 0.0 0.0 0:00.03 cpuhp/1
21 root -51 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/1
22 root rt 0 0 0 0 S 0.0 0.0 0:00.50 migration/1
23 root 20 0 0 0 0 S 0.0 0.0 0:01.89 ksoftirqd/1
25 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/1:0H-events_highpri
26 root 20 0 0 0 0 S 0.0 0.0 0:00.01 cpuhp/2
27 root -51 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/2
28 root rt 0 0 0 0 S 0.0 0.0 0:00.54 migration/2
29 root 20 0 0 0 0 S 0.0 0.0 0:01.24 ksoftirqd/2
31 root 0 -20 0 0 0 I 0.0 0.0 0:00.00 kworker/2:0H-events_highpri
32 root 20 0 0 0 0 S 0.0 0.0 0:00.00 cpuhp/3
33 root -51 0 0 0 0 S 0.0 0.0 0:00.00 idle_inject/3
34 root rt 0 0 0 0 S 0.0 0.0 0:00.55 migration/3
35 root 20 0 0 0 0 S 0.0 0.0 0:01.17 ksoftirqd/3

cardi~/Documents/vit/os/22bps1059 ls | grep trial
trialcopy.moon
cardi~/Documents/vit/os/22bps1059 ifconfig
Command 'ifconfig' not found, but can be installed with:
sudo apt install net-tools
[sudo] password for thecuber:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following package was automatically installed and is no longer required:
gir1.2-ayatanaapindicator3-0.1
Use 'sudo apt autoremove' to remove it.
The following NEW packages will be installed:
net-tools
0 upgraded, 1 newly installed, 0 to remove and 105 not upgraded.
Need to get 204 kB of archives.
After this operation, 819 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/main amd64 net-tools amd64 1.60+git20181103.0eebece-1ubuntu5 [204 kB]
Fetched 204 kB in 2s (83.8 kB/s)
Selecting previously unselected package net-tools.
(Reading database ... 255791 files and directories currently installed.)
Preparing to unpack .../net-tools_1.60+git20181103.0eebece-1ubuntu5_amd64.deb ...
Unpacking net-tools (1.60+git20181103.0eebece-1ubuntu5) ...
Setting up net-tools (1.60+git20181103.0eebece-1ubuntu5) ...
Processing triggers for man-db (2.18.2-1) ...
cardi~/Documents/vit/os/22bps1059 ifconfig
enp4s0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
      ether 54:05:db:a9:41:a1 txqueuelen 1000 (Ethernet)
      RX packets 0 bytes 0 (0.0 B)
      RX errors 0 dropped 0 overruns 0 frame 0
      TX packets 0 bytes 0 (0.0 B)
      TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
      inet 127.0.0.1 brd 255.0.0.0
          netmask 255.0.0.0
      lo6 0:0:1 prefixlen 128 scopeid 0x10<host>
          loop txqueuelen 1000 (Local Loopback)
          RX packets 97046 bytes 13293939 (13.2 MB)
          RX errors 0 dropped 0 overruns 0 frame 0
          TX packets 97046 bytes 13293939 (13.2 MB)
          TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

cardi~/Documents/vit/os/22bps1059 ping google.com
PING google.com (142.250.77.174) 56(84) bytes of data.
64 bytes from maa05s17-in-f14.1e100.net (142.250.77.174): icmp_seq=1 ttl=57 time=3.62 ms
64 bytes from maa05s17-in-f14.1e100.net (142.250.77.174): icmp_seq=2 ttl=57 time=5.28 ms
^C
--- google.com ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1002ms
rtt min/avg/max/mdev = 3.623/4.451/5.279/0.828 ms
cardi~/Documents/vit/os/22bps1059 ls
trialcopy.moon
cardi~/Documents/vit/os/22bps1059 scp trialcopy.moon trial.moon
cardi~/Documents/vit/os/22bps1059 ls
trialcopy.moon trial.moon
cardi~/Documents/vit/os/22bps1059 |

```

second

```
cardi~/Documents/vit/os$ cd 22bps1059/
cardi~/Documents/vit/os/22bps1059$ ls
trialcopy.moon trial.moon
cardi~/Documents/vit/os/22bps1059$ mkdir 22BPS1059
cardi~/Documents/vit/os/22bps1059$ mv trial.moon 22BPS1059/trial.moon
cardi~/Documents/vit/os/22bps1059$ ls
22BPS1059 trialcopy.moon
cardi~/Documents/vit/os/22bps1059$ cd 22BPS1059/
cardi~/Documents/vit/os/22bps1059/22BPS1059$ ls
trial.moon
cardi~/Documents/vit/os/22bps1059/22BPS1059$ cd ..
cardi~/Documents/vit/os/22bps1059$ ls
22BPS1059 trialcopy.moon
cardi~/Documents/vit/os/22bps1059$ |
```

third

```
1#!/bin/bash
2
3
4 touch filename.txt
5 ls
6 mkdir 22bps1059a
7 ls
8 cd 22bps1059a
9 ls
10 cd ..
11 mv filename.txt 22bps1059a/filename.txt
12 cd 22bps1059a
13 ls

cardi~/Documents/vit/os/22bps1059$ bash script.sh
22BPS1059 filename.txt script.sh trialcopy.moon
22BPS1059 22bps1059a filename.txt script.sh trialcopy.moon
filename.txt
```