

- **Slip:-5,14,26(FiFo)**

```
#include <stdio.h>

int main() {
    int ref[] = {3,4,5,6,3,4,7,3,4,5,6,7,2,4,6};
    int m = 15, n, frame[10], front = 0;
    int faults = 0, i, j, found;
    printf("Enter number of frames: ");
    scanf("%d", &n);
    for (i = 0; i < n; i++)
        frame[i] = -1;
    printf("\nPage\tFrames\t\tFault\n");
    for (i = 0; i < m; i++) {
        found = 0;
        for (j = 0; j < n; j++)
            if (frame[j] == ref[i])
                found = 1;
        if (!found) {
            frame[front] = ref[i];
            front = (front + 1) % n;
            faults++;
        }
        printf("%d\t", ref[i]);
        for (j = 0; j < n; j++)
            if (frame[j] == -1) printf("- ");
            else printf("%d ", frame[j]);
        printf("\t%s\n", found ? "No" : "Yes");
    }
    printf("\nTotal Page Faults = %d\n", faults);
    printf("Total Page Replacements = %d\n", faults > n ? faults - n : 0);
    return 0;
}
```

- **Output :-**

Enter number of frames: 3

Page	Frames	Fault
3	3 - -	Yes

4	3 4 -	Yes
5	3 4 5	Yes
6	6 4 5	Yes
3	6 3 5	Yes
4	6 3 4	Yes
7	7 3 4	Yes
3	7 3 4	No
4	7 3 4	No
5	7 5 4	Yes
6	7 5 6	Yes
7	7 5 6	No
2	2 5 6	Yes
4	2 4 6	Yes
6	2 4 6	No

Total Page Faults = 11

Total Page Replacements = 8

- **Slip: - 7,8,19,21(nice system call())**

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

```
#include <unistd.h>
```

```
#include <sys/types.h>
```

```
#include <sys/wait.h>
```

```
int main() {
```

```
    pid_t pid;
```

```

pid = fork();
if (pid < 0) {
    printf("Fork failed\n");
    return 1;
}
if (pid == 0) {
    printf("Child: Old nice value = %d\n", nice(0));
    printf("Child: Increasing nice value...\n");
    nice(5);
    printf("Child: New nice value = %d\n", nice(0));
}
else {
    printf("Parent process, PID = %d\n", getpid());
    wait(NULL);
}
return 0;
}

```

- **Output:-**

Parent process, PID = 2451

Child: Old nice value = 0

Child: Increasing nice value...

Child: New nice value = 5

- **Slip:-9,13,17,23 (non-preemptive priority scheduling sjf)**

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

```
#include <stdlib.h>
```

```

#include <time.h>

struct P {
    int id, at, bt, wt, tat, done;
};

int main() {
    int n, i, t = 0, c = 0, m;
    float aw = 0, at = 0;
    srand(time(0));
    printf("Enter number of processes: ");
    scanf("%d", &n);
    struct P p[n];
    for(i = 0; i < n; i++) {
        p[i].id = i + 1;
        p[i].done = 0;
        printf("Enter Arrival Time and First CPU Burst for P%d: ", i + 1);
        scanf("%d%d", &p[i].at, &p[i].bt);
        p[i].bt += rand() % 5;
    }
    printf("\nGantt Chart:\n");
    while(c < n) {
        m = -1;
        for(i = 0; i < n; i++)
            if(!p[i].done && p[i].at <= t &&
                (m == -1 || p[i].bt < p[m].bt))
                m = i;
        if(m == -1) {
            t++;
            continue;
        }
    }
}

```

```

    }

    printf(" | P%d ", p[m].id);

    p[m].wt = t - p[m].at;

    t += p[m].bt;

    p[m].tat = t - p[m].at;

    p[m].done = 1;

    c++;

}

printf("\n\nProcess\tAT\tBT\tWT\tTAT\n");

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

    printf("P%d\t%d\t%d\t%d\t%d\n",

        p[i].id, p[i].at, p[i].bt, p[i].wt, p[i].tat);

    aw += p[i].wt;

    at += p[i].tat;

}

printf("\nAverage Waiting Time = %.2f\n", aw/n);

printf("Average Turnaround Time = %.2f\n", at/n);

return 0;

}

```

• Output:-

Enter number of processes: 4

Enter Arrival Time and First CPU Burst for P1: 0 5

Enter Arrival Time and First CPU Burst for P2: 1 3

Enter Arrival Time and First CPU Burst for P3: 2 2

Enter Arrival Time and First CPU Burst for P4: 3 4

Gantt Chart:

| P1 | P3 | P2 | P4 |

Process	AT	BT	WT	TAT
P1	0	8	0	8
P2	1	6	9	15
P3	2	4	6	10
P4	3	7	10	17

Average Waiting Time = 6.25

Average Turnaround Time = 12.50

- **Slip:-10,24,30(round robin):-**

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
typedef struct {
    int id,at,bt,rem,wt,tat;
} P;
int main() {
    int n,q,t=0,d=0,i,run;
    float aw=0,aa=0;
    srand(time(0));

    printf("Enter number of processes: ");
    scanf("%d",&n);
    P p[n];
    for(i=0;i<n;i++) {
        p[i].id=i+1;
        printf("Enter Arrival Time and First CPU Burst for P%d: ",i+1);
        scanf("%d%d",&p[i].at,&p[i].bt);
        p[i].bt+=rand()%5;
        p[i].rem=p[i].bt;
    }
    printf("Enter Time Quantum: ");
    scanf("%d",&q);
```

```

printf("\nGantt Chart:\n|");
while(d<n) {
    run=0;
    for(i=0;i<n;i++)
        if(p[i].rem && p[i].at<=t) {
            run=1;
            printf(" P%d |",p[i].id);
            if(p[i].rem>q)
                t+=q,p[i].rem-=q;
            else {
                t+=p[i].rem;
                p[i].rem=0;
                p[i].tat=t-p[i].at;
                p[i].wt=p[i].tat-p[i].bt;
                d++;
            }
        }
    if(!run) t++;
}

printf("\n\nProcess\tAT\tBT\tWT\tTAT\n");
for(i=0;i<n;i++) {
    printf("P%d\t%d\t%d\t%d\t%d\n",
        p[i].id,p[i].at,p[i].bt,p[i].wt,p[i].tat);
    aw+=p[i].wt; aa+=p[i].tat;
}

printf("\nAverage Waiting Time = %.2f\n",aw/n);
printf("Average Turnaround Time = %.2f\n",aa/n);
return 0;
}

```

- **Output:-**

Enter number of processes: 4
Enter Arrival Time and First CPU Burst for P1: 0 5
Enter Arrival Time and First CPU Burst for P2: 1 3
Enter Arrival Time and First CPU Burst for P3: 2 4
Enter Arrival Time and First CPU Burst for P4: 3 2
Enter Time Quantum: 2

Gantt Chart:

| P1 | P2 | P3 | P4 | P1 | P2 | P3 | P4 |

Process	AT	BT	WT	TAT
---------	----	----	----	-----

P1	0	7	8	15
----	---	---	---	----

P2	1	5	7	12
----	---	---	---	----

P3	2	6	9	15
----	---	---	---	----

P4	3	4	9	13
----	---	---	---	----

Average Waiting Time = 8.25

Average Turnaround Time = 13.75

Slip :-16,20,27(LRU replacement Algorithm)

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

```
int main() {
```

```
    int r[]={3,5,7,2,5,1,2,3,1,3,5,3,1,6,2};
```

```
    int f[10],c[10],n,i,j,k,t=0,p,pos,fault=0;
```

```
    printf("Enter number of frames: ");
```

```
    scanf("%d",&n);
```

```
    for(i=0;i<n;i++) f[i]=c[i]=-1;
```

```
    printf("\nPage\tFrames\t\tStatus\n");
```

```
    for(i=0;i<15;i++) {
```

```
        p=r[i]; t++; pos=-1;
```

```
        for(j=0;j<n;j++)
```

```
            if(f[j]==p) pos=j;
```

```
        if(pos==-1) {
```

```
            fault++;
```

```
            pos=0;
```

```
            for(j=0;j<n;j++)
```

```
                if(f[j]==-1) { pos=j; break; }
```

```
            if(f[pos]!=-1)
```

```
                for(j=1;j<n;j++)
```

```
                    if(c[j]<c[pos] && f[j]!=-1) pos=j;
```

```

        f[pos]=p;
        printf("%d\t",p);
        for(k=0;k<n;k++) printf("%d ",f[k]);
        c[pos]=t;
        printf("\tFault\n");
    } else {
        c[pos]=t;
        printf("%d\t",p);
        for(k=0;k<n;k++) printf("%d ",f[k]);
        printf("\tHit\n");
    }
}

printf("\nTotal Page Faults = %d\n",fault);

return 0;
}

```

• Output:-

Enter number of frames: 3

Page	Frames	Status
3	3 - -	Fault
5	3 5 -	Fault
7	3 5 7	Fault
2	2 5 7	Fault
5	2 5 7	Hit
1	2 5 1	Fault
2	2 5 1	Hit

3	2 3 1	Fault
1	2 3 1	Hit
3	2 3 1	Hit
5	5 3 1	Fault
3	5 3 1	Hit
1	5 3 1	Hit
6	5 6 1	Fault
2	2 6 1	Fault

Total Page Faults = 9

- **Slip:-12,29(FCFS CPU scheduling)**

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
typedef struct{int id,at,bt,wt,tat;}P;
int main(){
    int n,i,t=0; float aw=0,aa=0;
    srand(time(0));
    printf("Enter number of processes: ");
    scanf("%d",&n);
    P p[n];
    for(i=0;i<n;i++){
        p[i].id=i+1;
        printf("Enter Arrival Time and First CPU Burst for P%d: ",i+1);
        scanf("%d%d",&p[i].at,&p[i].bt);
        p[i].bt+=rand()%5;
    }
    for(i=0;i<n;i++){
        if(t<p[i].at)t=p[i].at;
        p[i].wt=t-p[i].at;
        t+=p[i].bt;
        p[i].tat=t-p[i].at;
    }
}
```

```

printf("\nGantt Chart:\n|");
for(i=0;i<n;i++)printf(" P%d |",p[i].id);
printf("\n\nProcess\tAT\tBT\tWT\tTAT\n");
for(i=0;i<n;i++){
    printf("P%d\t%d\t%d\t%d\t%d\n",
        p[i].id,p[i].at,p[i].bt,p[i].wt,p[i].tat);
    aw+=p[i].wt; aa+=p[i].tat;
}
printf("\nAverage Waiting Time = %.2f\n",aw/n);
printf("Average Turnaround Time = %.2f\n",aa/n);
return 0;
}

```

• Output:-

Enter number of processes: 4
Enter Arrival Time and First CPU Burst for P1: 0 5
Enter Arrival Time and First CPU Burst for P2: 1 3
Enter Arrival Time and First CPU Burst for P3: 2 4
Enter Arrival Time and First CPU Burst for P4: 3 2

Gantt Chart:

| P1 | P2 | P3 | P4 |

Process	AT	BT	WT	TAT
P1	0	7	0	7
P2	1	5	6	11
P3	2	8	10	18
P4	3	6	18	24

Average Waiting Time = 8.50

Average Turnaround Time = 15.00