$22bps1059$

$first - sigint$

File Edit View Search Terminal Help

```c
#include <stdio.h>
#include <signal.h>
#include <sys/wait.h>
#include <unistd.h>
#include <stdlib.h>
#define SIGINT 2

void  siginthandler() {
    printf("handled\n");
    exit(0);
}

int main() {
    signal(SIGINT,siginthandler);
    while(1) {
        printf("Hello world\n");
        sleep(1);
    }
    return 0;
}
```

```
22BPS1059>./sigint2
Hello world
Hello world
Hello world
Hello world
Hello world
Hello world
Hello world
Hello world
^Chandled
22BPS1059>
```

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

```c
#include <signal.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>
void sighup(); void sigint(); void sigquit();

void main() {
    int pid;
    pid = fork();
    if(pid == 0) {
        signal(SIGHUP,sighup);
        signal(SIGQUIT,sigint);
        signal(SIGINT,sigint);
        signal(SIGQUIT,sigquit);

        for(;;);
    }
    else {
        printf("\n parent: sending sighup\n\n");
        kill(pid,SIGHUP);
        sleep(3);
        printf("\n parent: sending sigint\n\n");
        kill(pid,SIGINT);
        sleep(3);
        printf("\n parent: sending sigquit\n\n");
        kill(pid,SIGQUIT);
        sleep(3);
    }
}

void sighup() {
    printf("CHILD: i have recieved a SIGHUP\n");
    fflush(stdout);
}

void sigint() {
    printf("CHILD: i have recieved a sigint\n");
    fflush(stdout);
}

void sigquit() {
    printf("CHILD: i have recieved a sigquit\n");
    fflush(stdout);
}
```

```
22BPS1059>./sigint

 parent: sending sighup

CHILD: i have recieved a SIGHUP

 parent: sending sigint

CHILD: i have recieved a sigint

 parent: sending sigquit

CHILD: i have recieved a sigquit
22BPS1059>
```

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

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/wait.h>
#define MAX_PROCESSES 5
#define TIME_QUANTUM 2
int completed = 0;
void execute(int pid, int time_slice) {
    printf("Executing Process %d for %d seconds\n", pid, time_slice);
    sleep(time_slice);
}
int main() {
    int n = 5; // Number of processes
    int arrival_time[] = {0, 1, 2, 3, 4}; /* W: unused variable 'arrival_time' [-Wunused-
    int burst_time[] = {6, 2, 8, 3, 5};
    int remaining_time[MAX_PROCESSES];
    int child_pids[MAX_PROCESSES];

    for (int i = 0; i < n; i++) {
        remaining_time[i] = burst_time[i];
    }

    int current_process = 0; /* W: unused variable 'current_process' [-Wunused-variable]
    for (int i = 0; i < n; i++) {
        pid_t child_pid = fork();

        if (child_pid == -1) {
            perror("fork");
            exit(1);
        }
        else if (child_pid == 0) {
            while (remaining_time[i] > 0) {
                int time_slice = (remaining_time[i] < TIME_QUANTUM) ? remaining_time[i] :
 TIME_QUANTUM;
                execute(i + 1, time_slice);
                remaining_time[i] -= time_slice;

                if (remaining_time[i] > 0) {
                    printf("Process %d preempted. Remaining time: %d seconds\n", i + 1, r
emaining_time[i]);
                }
            }
            exit(0);
        }
        else {
            child_pids[i] = child_pid;
        }
    }
    for (int i = 0; i < n; i++) {
        waitpid(child_pids[i], NULL, 0);
        printf("Parent Process: Child Process %d completed.\n", child_pids[i]);
    }

}
```

roundrobin.c 14,9 Top

"roundrobin.c" 57L, 1552B written

```
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 [RO] 45,0-1 93%

[0] 0:bash* "thecuber-ThinkPad-E14" 21:37 05-Sep-23

$fourth - srjf$

File Edit View Search Terminal Help

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/wait.h>
#define MAX_PROCESSES 5
int completed = 0;

void execute(int pid, int time_slice) {
    printf("Executing Process %d for %d seconds\n", pid, time_slice);
    sleep(time_slice);
}
int main() {
    int n = 5;
    int arrival_time[] = {0, 1, 2, 3, 4};
    int burst_time[] = {6, 2, 8, 3, 5};
    int remaining_time[MAX_PROCESSES];
    int child_pids[MAX_PROCESSES];
    for (int i = 0; i < n; i++) remaining_time[i] = burst_time[i];
    int current_time = 0;

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

        if (child_pid == -1) {
            perror("fork");
            exit(1);
        } else if (child_pid == 0) { // Child process
            while (remaining_time[i] > 0 && !completed) {
                int time_slice = 1;
                if (remaining_time[i] < 1) {
                    time_slice = remaining_time[i];
                }

                execute(i + 1, time_slice);
                remaining_time[i] -= time_slice;
            }

            exit(0);
        } else { // Parent process
            child_pids[i] = child_pid;
        }
    }
    sleep(2);
    for (int i = 0; i < n; i++) {
        kill(child_pids[i], SIGHUP);
    }
    for (int i = 0; i < n; i++) {
        int status;
        waitpid(child_pids[i], &status, 0);
        if (WIFSIGNALED(status) && WTERMSIG(status) == SIGHUP) {
            printf("Parent Process: Child Process %d terminated.\n", child_pids[i]);
        }
    }
    return 0;
}
```

srjf.c 21,25 Top

"srjf.c" 58L, 1587B written

srjf.c [RO] 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>
```

[0] 0:bash* "thecuber-ThinkPad-E14" 21:39 05-Sep-23