

Create thread using C

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

```
Void *funThread (void *arg);
```

```
int main () {
```

```
    pthread_t t1;
```

```
    pthread_create (&t1, NULL, funThread, NULL);
```

```
    pthread_join (t1, NULL);
```

```
    return 0;
```

```
}
```

```
Void *funThread (void *arg) {
```

```
    printf ("Enter thread: \n");
```

```
    for (int i=0; i<3; i++) {
```

```
        printf ("thread: %d \n", i);
```

```
}
```

```
    printf ("Done with thread... \n");
```

```
}
```

Output

Entered thread:

Done with thread.

variable memory address

name of function

parameters to the function

always null

Thread go start run

Parameter pass run

Fourth parameter

memory address pass

run

act ratio 20/60

ESF 32

$$\text{Elevated} \rightarrow 80 \times 19 = 1520 \text{ board ft}$$

$$\text{BRAC} \rightarrow 50 \times 4 = 200 \text{ board ft}$$

$$\text{Parking} \rightarrow 150 \text{ board ft}$$

$$\text{Parking} \rightarrow 50 \text{ board ft}$$

$$\text{Parking} \rightarrow 60 \text{ board ft}$$

$$\text{Total : } 1980 \text{ board ft}$$

$$+ 50 \text{ board ft}$$

$$+ 80 \text{ board ft}$$

$$\text{Hut } 2000$$

ese 420 ✓

HUM 101 (ex) board ft + board ft

EDS DL board ft

ANT 101 (ex) board ft

morning direct (101)

mat 215 (ex) board ft

??? LSF (425) direct end (101) board ft

high

board ft

board ft

Thread 9 pointers
input $\underline{\text{int}}$ + pointers
output $\underline{\text{int}}$.

```
int * func_thread (int, *v);
```

```
int * t - ret;
```

```
int main () {
```

```
    pthread_t t1; (i) memory address
```

```
    int n = 5;
```

```
    pthread_create (&t1, NULL, func_thread, &n);
```

```
    pthread_join (t1, &t - ret);
```

```
    printf ("Thread returned : %d\n", *t - ret);
```

```
    return 0;
```

(ii)

```
}
```

```
int * func_thread (int *v) {
```

```
    *v = *v * 5;
```

```
    return *v;
```

as int receive
pointer pass

pointer go Bhutan value 5 digit
multiply $\underline{\underline{25}}$, $5 \times 5 = 25$.

Output:

Thread returned 25.

$*v$ = pointer \rightarrow address.

v = value.

T1

Context switch can happen at any time.

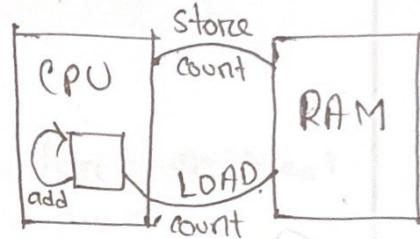
Count =

Count + 1

i) Load

ii) Add

iii) Store



(iii)

T1 T2 (i)

Count: 1 Count: 2

Count: 2

Count: 2 position ←

Program value stored in memory weird behaviour

Program value in memory is lost chance to

put value back

if we can control the context switch then, we solve the problem.

Subject _____

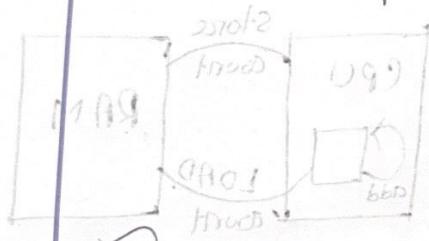
Sat Sun Mon Tue Wed Thu Fri
○ ○ ○ ○ ○ ○ ○

Date : / /

multiple Processes Synchronization

Goal To implement

When multiple process manipulates a variable,
a condition prevails, called Race condition.



b001 (i)

b0A (ii)

note (iii)

section ১৪ গ্রাম কন্টেক্স্ট স্বিচ না

ক্রাইটিক্যাল সেকশন, অর্থাৎ সেকশন অফ কোড (১) অর্থাৎ

critical section.

S : fruo) L : fruo)

S : fruo)

Sub

Load

add

Store

→ critical section.

allowed only one

thread at a time.

↳ called mutually

exclusion.

multiple threads can't access the same resource simultaneously

Subject _____

Sat Sun Mon Tue Wed Thu Fri
○ ○ ○ ○ ○ ○ ○

Date: / /

When a critical section is empty and a thread wants to enter the critical section, we should let it enter. This is called a progress.

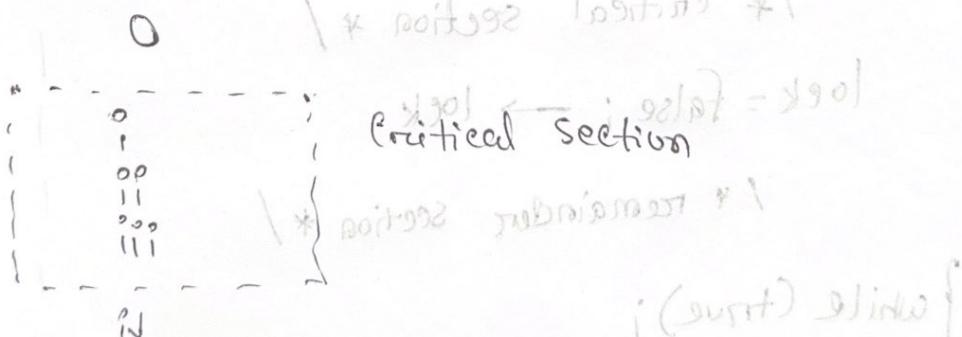
Hardware based solution for critical section problem:

Counter ++

Counter --

पूर्ण रूप से क्रिटिकल सेक्शन.

TEST-AND-SET()



boolean test-and-set (boolean *target) {

 atomic
 {
 boolean ret = *target;
 *target = true;
 }

 return ret;

atomic

in nature.

means, we
cannot break

→ it returns the previous value.

the function.

→ make it true forcefully.

And, after

in this
function.

context +
switch
cannot

happen.

Either it

will run
or it won't.

Hardware,

ensures that
context switch

won't happen

here in the

atomic function.

do {

 while (test-and-set (lock)) → entry;

 /* do nothing */

 /* critical section */

 lock = false; → lock

 /* remainder section */

}

When a critical section is empty (topnotch) and a thread wants to enter the critical section, we should let it enter. This is called priority progress.

Priority Progress.

another ni

new address

second form

Hardware based solution for critical section problem:

Test-and-set

Counter ++

Counter --

92

पूर्णी रखे critical section.

? ob

Test-AND-SET()

* position ob *

int result

out result

get_pos()

set_pos()

! simots

0

* position * |

920 ← ; get_pos = 920

critical section

N

i (exit) link {

63

Subject

mutual exclusion

Sat	Sun	Mon	Tue	Wed	Thu	Fri
○	○	○	○	○	○	○

Date : / /

$T_1, T_2 \rightarrow \text{while } (\text{TAS}(\&\text{lock})) \{ \dots \}$.

(i)
progress

$T_1 \rightarrow$

- { i) Load
- ii) Add
- iii) Store

$T_1 \rightarrow \text{lock} = \text{False}$

lock = False.

True.
false

Iterating in an empty while loop is called spinning.

$\text{while } (\text{true}) \rightarrow \text{infinite while loop.}$