Lecture-13
BDU

Date 19/09/18
Page

lock (global) → False (initially)
key (local) → False

```
do{
    key = true;                    → entry section
    while (key == true)
        swap (lock, key);
    Critical section
    lock = false                   → exit section
    Remainder section
} while (1);
```

Atomicity -
Another process can use test & set only after one is done with it.

lock	P_i	P_j
F	key = ~~F~~ ~~T~~ F	key = ~~F~~ ~~T~~ F
~~T~~ ~~F~~ T		

```
boolean Testandset (*target)
{
    boolean rv = *target;
    *target = true;
    return rv;
}
```

key	lock	P_0	P_1	P_2
F	F	W(0) = ~~F~~	W(1) = ~~F~~	W(2) = F
~~T~~ F		~~T~~ F	T	
~~T~~ F				

boolean waiting [n] = local
" lock = global
" key = global

```
do {
    waiting [i] = true;                         → entry
    key = true;
    while ( waiting [i] && key)
        key = Test and set (lock);
    waiting [i] = false
    critical section;
    j = (i+1) % n;                              → exit
    while ( (j != i) && !waiting [j])
        j = (j+1) % n;
    if (j == i) lock = false;
    else waiting [j] = false;
    remainder section;
} while(1)
```

Semaphores

S → an integer variable accessed only through two atomic operation

wait (S) – P (to test)

signal (S) – V (to increment)

```
signal (S)
{
    S++;
}
```

```
wait (S)
{
    while (S<=0);
        S--;
}
```

```
do {
    wait (mutex);
    critical section;
    signal (mutex);
    remainder section.
} while (1);
```

mutex = 1 0 1 0