Date 20/4

PROCESS SYNCHRONIZATION

The Producer Consumer Problem

Note that **counter++;** **← this line is NOT what it seems!!** Imp.

is really --> register = counter
register = register + 1
counter = register

Counter ++ micro-Instruction

At a micro level, the following scenario could occur using this code:

TO;	**Producer**	**Execute**	**register1 = counter**	**register1 = 5**
T1;	**Producer**	**Execute**	**register1 = register1 + 1**	**register1 = 6**
T2;	**Consumer**	**Execute**	**register2 = counter**	**register2 = 5**
T3;	**Consumer**	**Execute**	**register2 = register2 - 1**	**register2 = 4**
T4;	**Producer**	**Execute**	**counter = register1**	**counter = 6**
T5;	**Consumer**	**Execute**	**counter = register2**	**counter = 4**

PROCESS SYNCHRONIZATION

Critical Sections

A section of code, common to n cooperating processes, in which the processes may be accessing common variables.

A Critical Section Environment contains:

Entry Section Code requesting entry into the critical section.

Critical Section Code in which only one process can execute at any one time.

Exit Section The end of the critical section, releasing or allowing others in.

Remainder Section Rest of the code AFTER the critical section.

PROCESS SYNCHRONIZATION

Critical Sections

The critical section must ENFORCE ALL THREE of the following rules:

Mutual Exclusion: No more than one process can execute in its critical section at one time.

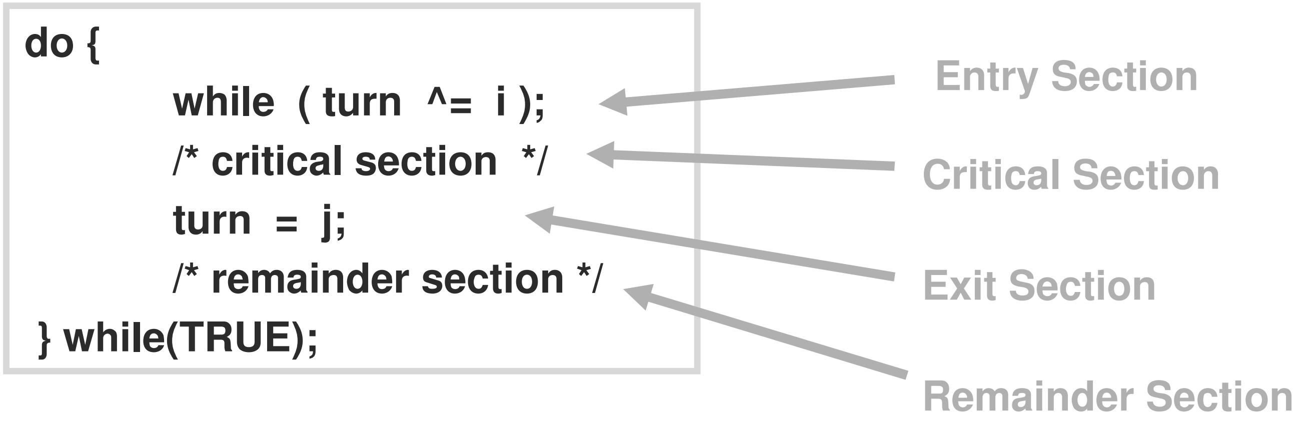
Progress: If no one is in the critical section and someone wants in, then those processes not in their remainder section must be able to decide in a finite time who should go in.

Bounded Wait: All requesters must eventually be let into the critical section.

PROCESS SYNCHRONIZATION

Two Processes Software

Here’s an example of a simple piece of code containing the components required in a critical section.



```
do {
          while  ( turn  ^=  i );
          /* critical section  */
          turn  =  j;
          /* remainder section */
 } while(TRUE);
```

Entry Section

Critical Section

Exit Section

Remainder Section

PROCESS SYNCHRONIZATION

Two Processes Software

Here we try a succession of increasingly complicated solutions to the problem of creating valid entry sections.

NOTE: In all examples, **i** is the current process, **j** the "other" process. In these examples, envision the same code running on two processors at the same time.

TOGGLED ACCESS:

```
do {
            while  ( turn  ^=  i );
            /* critical section  */
            turn  =  j;
            /* remainder section */
 } while(TRUE);
```

Algorithm 1

Are the three Critical Section Requirements Met?