1.many to one many thread to one kernel

one to one one thread to one kernel

many to many many thread to many kernels

one to one is worst

2.cause they are lighter than process. The thread shares the same virtual memory space. Process is in the memory.

3.faster, the process has no common part that there is no need to share

4.a.1. B first

4.a.2 A first then C

4.a.3 a first then b then c

4.b.1 process b first/ all a then all b

4.b.2 process b first/no

5. no , we must have enough space for active page for other processes. If it is possible, then the os will be useless because no program can run by it.

6.

semaphore s1 = 0, s2 = 0,s3=0;

process P1{

<phase 1>

v(s1)

p(s3

<phase 2>

}

process P2

<phase 1>

v(s2)

p(s1)

<phase 2>

}

process P3

<phase 1>

v(s3)

p(s2)

<phase 2>

}

b/

semaphore s1 = 0, s2 = 0,s3=0,s4=0

process P1{

<phase 1>

v(s1)

p(s3)

<phase 2>

v(s4)

}

process P2

<phase 1>

v(s2)

p(s1)

p(s4)

<phase 2>

v(s5)

}

process P3

<phase 1>

v(s3)

p(s2)

p(s5)

<phase 2>

}

7/ cause the value are shared between differrent processes. If they are not CS, it can not lock.

8/ security problems. If a program can enter the main memory .It can steal other program's information