

A hand is shown typing on a keyboard, with a digital, wireframe background. The word "DEADLOCK" is written in large, yellow, bold, sans-serif capital letters across the center of the image. The overall color scheme is dark blue and black, with the yellow text providing a strong contrast.

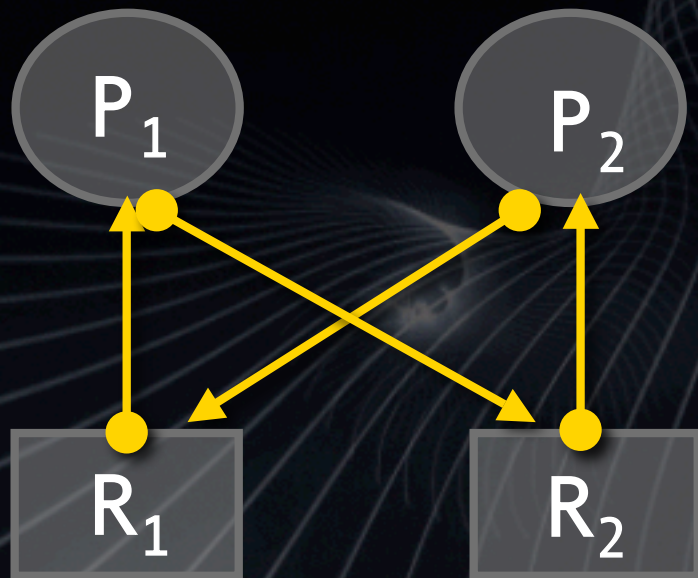
DEADLOCK

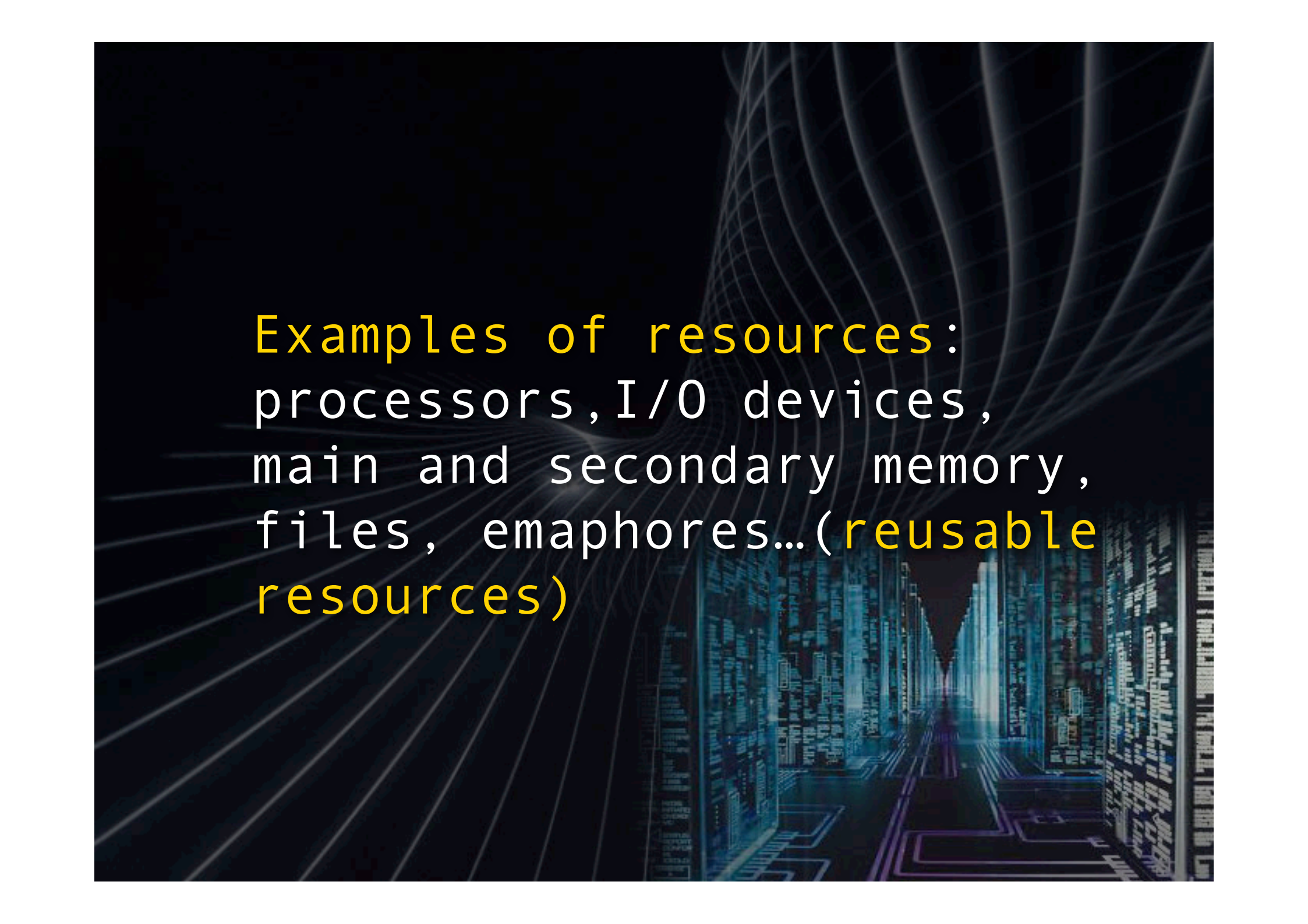
Contents

- Principles of deadlock
- Deadlock prevention
- Deadlock detection

Deadlock

A set of processes is deadlocked if each process in the set is waiting for an event that only another process in the set can cause.



The background of the slide is a dark, futuristic digital tunnel. It features glowing white and blue lines that curve and converge towards the center, creating a sense of depth and movement. On the right side, there are vertical panels that look like data screens or server racks, displaying various lines of code and information in a light blue font. The overall aesthetic is high-tech and digital.

Examples of resources:
processors, I/O devices,
main and secondary memory,
files, emaphores... (reusable
resources)

Utilization protocol

request

(the process can
be blocked)

use

release

The background of the slide features a close-up photograph of a hand holding a RAM module. The hand is positioned at the top, with fingers gripping the edges of the module. The RAM module is green and has several black chips on it. A white label with the word "ENTSPY" in blue is attached to the module. The entire scene is overlaid with a dark, semi-transparent grid pattern that creates a sense of depth and perspective, as if looking through a lens or a window. The lighting is dramatic, with highlights on the hand and the RAM module against a dark background.

P1

P(mutex1);

<R1>;

P(mutex2);

<R2>;

V(mutex2);

<release of R2>;

V(mutex1);

<release of R1>;

The background of the slide features a close-up, low-angle shot of a person's hand holding a green printed circuit board (PCB). The hand is positioned at the top, with fingers gripping the board. The PCB is populated with various electronic components, including several integrated circuits (chips) and a prominent white label with the word 'ENTSPY' printed in blue. The lighting is dramatic, with strong highlights and deep shadows, creating a sense of depth and focus on the hardware. The overall aesthetic is technical and modern.

P2

P(mutex2);

<R2>;

P(mutex1);

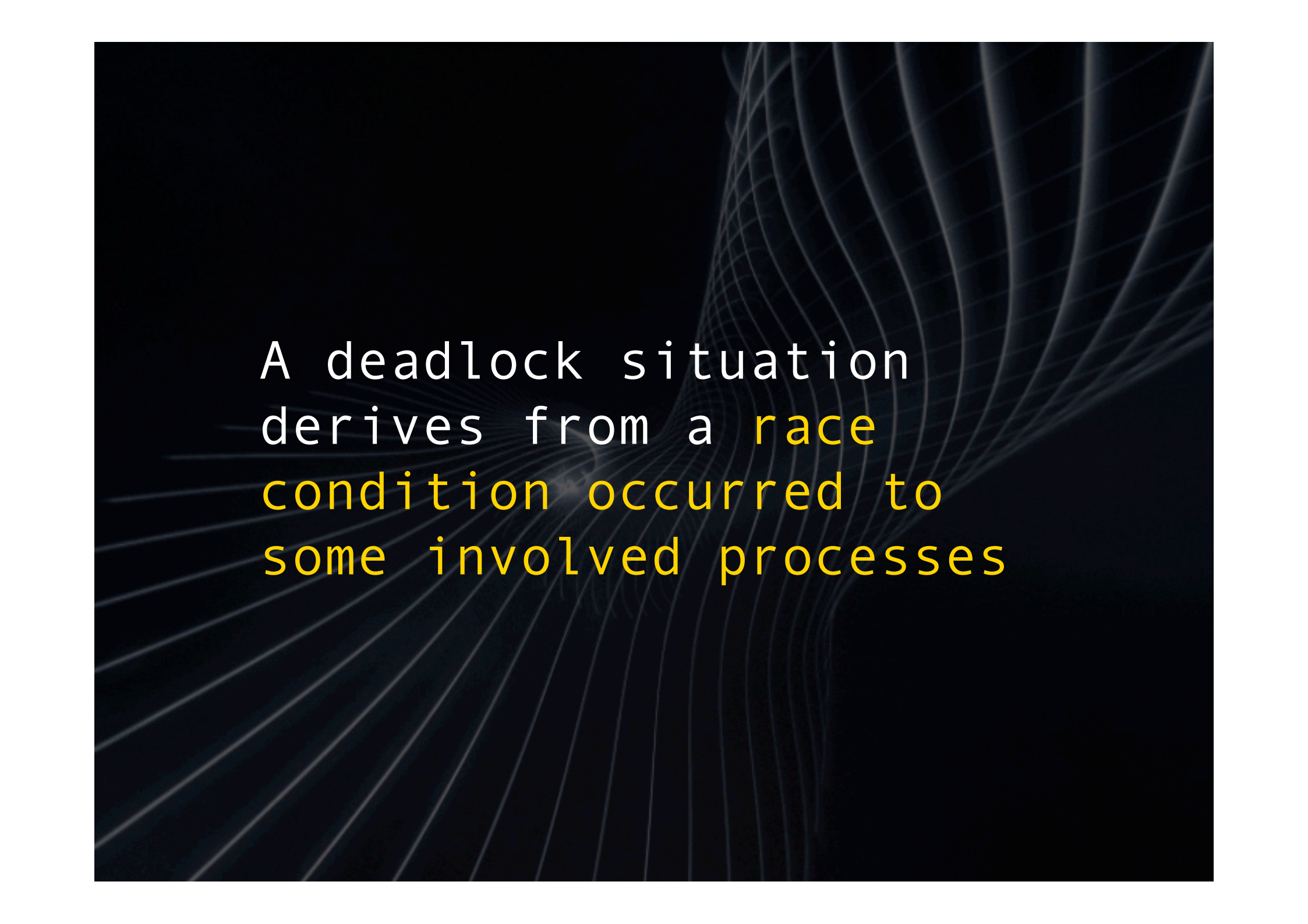
<R1>;

V(mutex1);

<release of R1>;

V(mutex2);

<release of R2>;



A deadlock situation
derives from a **race**
condition occurred to
some involved processes

Conditions for deadlock

P_1, P_2, \dots, P_n :

a set of processes

R_1, R_2, \dots, R_m :

a set of resource types

A deadlock situation can arise if the following four conditions hold at the same time:

```
total memory = 4096000, 3472000 used, 624000 free, 4096000 total
free memory = 624000, 492800 data, 131200 locked,
total swapped = 1120 + 14720 private, 26240 shared,
1420 active, 2780 inactive, 1000000 used, 320 free,
1000000 pages, 20140000 pageouts

CPU  TIME  #TH #PRTS #PROCS #PVTS #SWAP #PAGE #STCK #VIRT
0.0% 0:00.00 1 20 71 304K 2720K 1920K 3432
0.0% 0:00.00 1 14 19 256K 624K 800K 768
0.0% 0:00.00 1 24 55 240K 240K 720K 720
0.0% 0:21.24 1 21 29 2832K 180K 2520K 768
0.0% 0:00.21 2 23 73 996K 2360K 5140K 2448
0.0% 0:00.01 1 14 19 240K 624K 800K 768
0.0% 0:00.01 1 14 55 240K 240K 720K 720
0.0% 0:00.13 4 53 30 644K 2220K 1940K 872
0.0% 0:13.04 7 189 185 4240K 112 176 4192
0.0% 0:12.62 22 282 768 648 312 912 5072
0.1% 0:20.49 15 311 585 456 256 756 5184
0.1% 0:30.00 14 204 724 1432 180 1740 6272
0.0% 0:00.45 5 36 182 2000K 3120K 4140K 1808
0.0% 0:30.24 14 143 256 896K 6840K 220 440
0.0% 0:00.21 3 24 25 20K 272K 194K 768
0.0% 0:00.01 1 19 21 40K 104K 240K 768
0.0% 0:00.06 2 81 58 104K 140K 1200K 832
0.0% 0:00.23 2 25 28 64K 176K 272K 768
0.0% 2:29.29 20 262 919 1816 456 1872 13200
0.0% 0:00.14 1 58 83 104K 2276K 940K 3120
0.0% 0:00.09 1 49 42 124K 2112K 600K 2464
0.0% 0:01.38 2 99 81 1244K 404K 1000K 1104
0.0% 0:01.21 6 236 215 1272K 9452K 224K 4096
```

A hand is shown typing on a keyboard, with a blue glow emanating from the keys. The background is dark with abstract white lines that create a sense of motion and depth. The text is overlaid on the left side of the image.

mutual exclusion

hold-and-wait

no preemption

circular wait

The background features a dark blue gradient with a complex pattern of glowing, thin, light blue lines that curve and flow across the frame. A bright, circular glow is centered in the lower-left quadrant, from which the lines appear to emanate or be drawn towards. The overall effect is a sense of dynamic movement and depth.

All four conditions must
hold for deadlock to occur

System resource allocation graph

vertices:

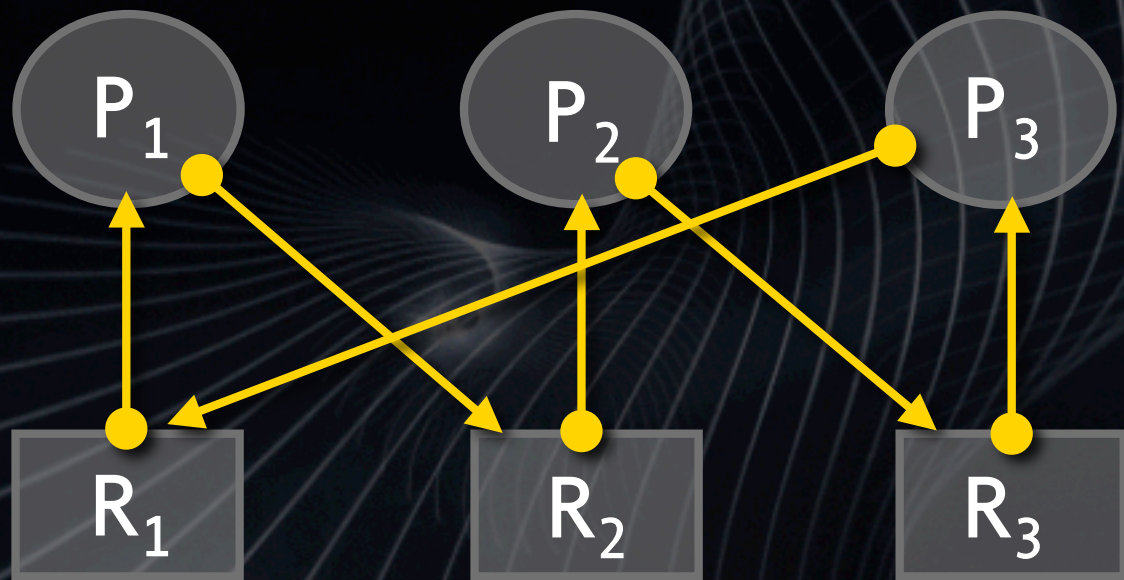
$P = (P_1, P_2, \dots, P_n)$

$R = (R_1, R_2, \dots, R_m)$

edges:

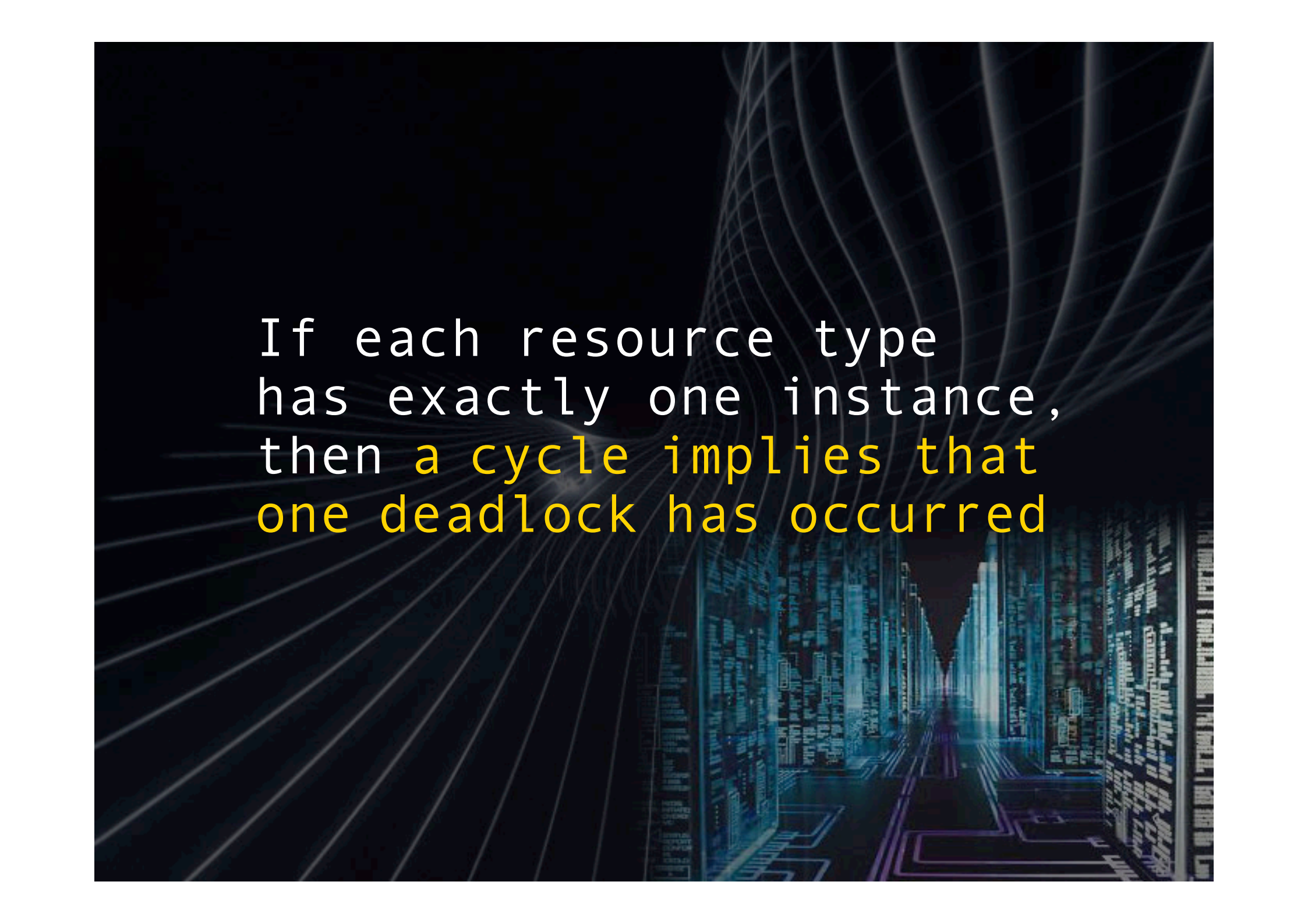
request edge $P_i \rightarrow R_j$

assignment edge $R_j \rightarrow P_i$

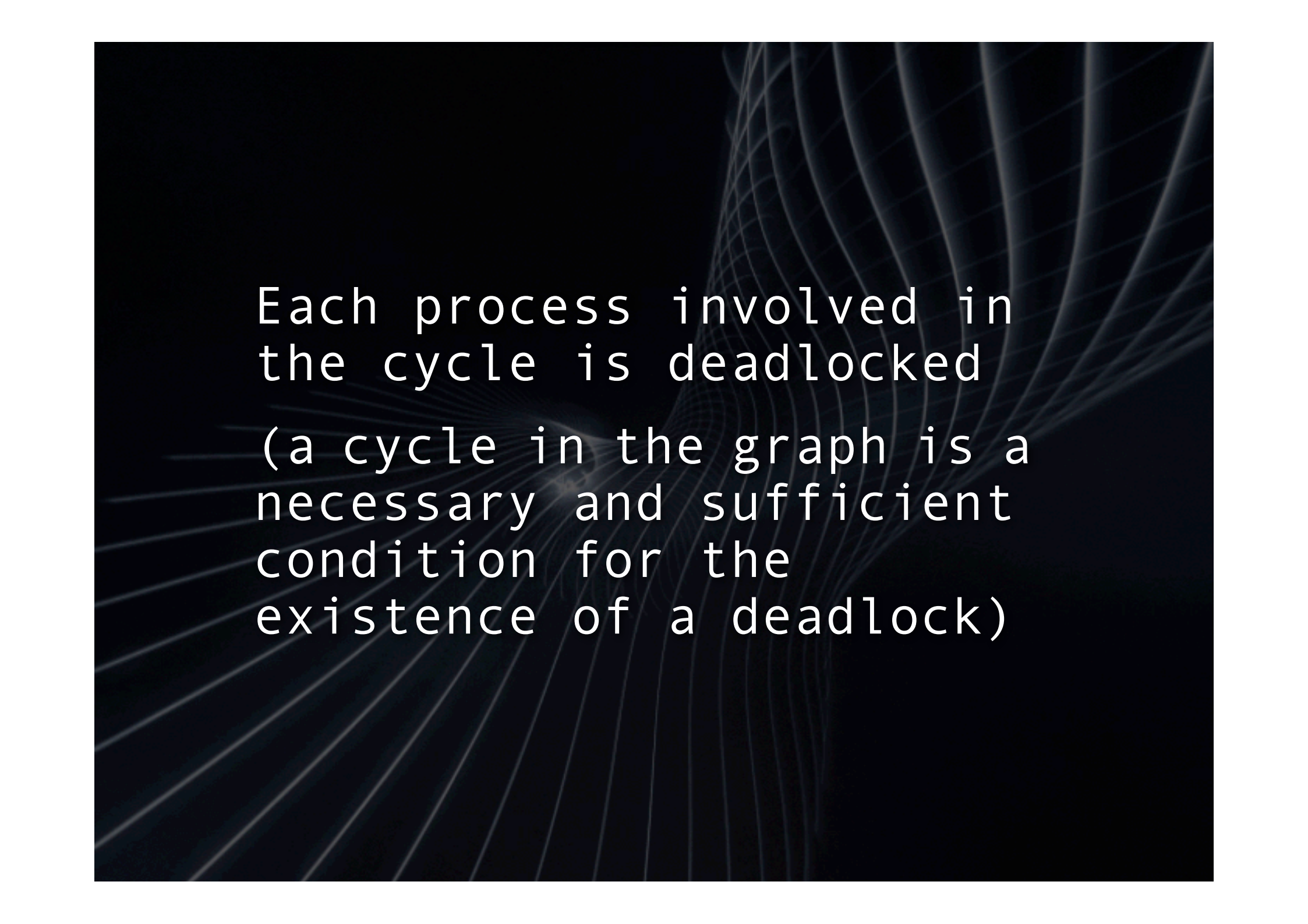


If the graph does not
contains cycles, then **no
process is deadlocked**

If the graph contain one
cycle, then a deadlock
may exist

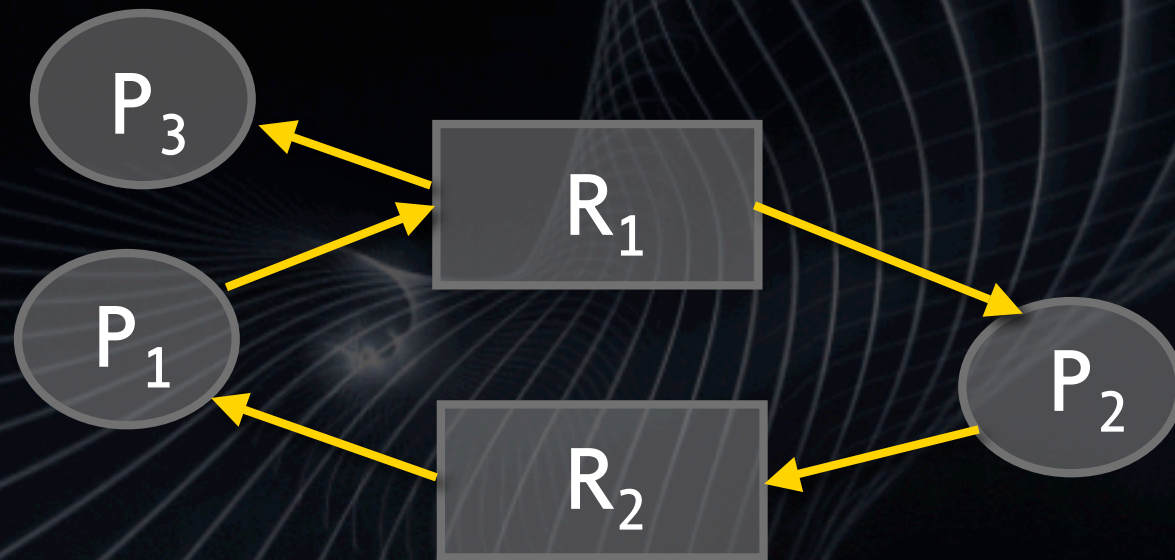
The background of the slide is a dark, futuristic digital environment. It features a perspective view of a tunnel formed by glowing, curved lines that recede into the distance. On the right side, there are vertical panels displaying streams of data or code in a light blue color. The overall aesthetic is high-tech and abstract.

If each resource type
has exactly one instance,
then a cycle implies that
one deadlock has occurred



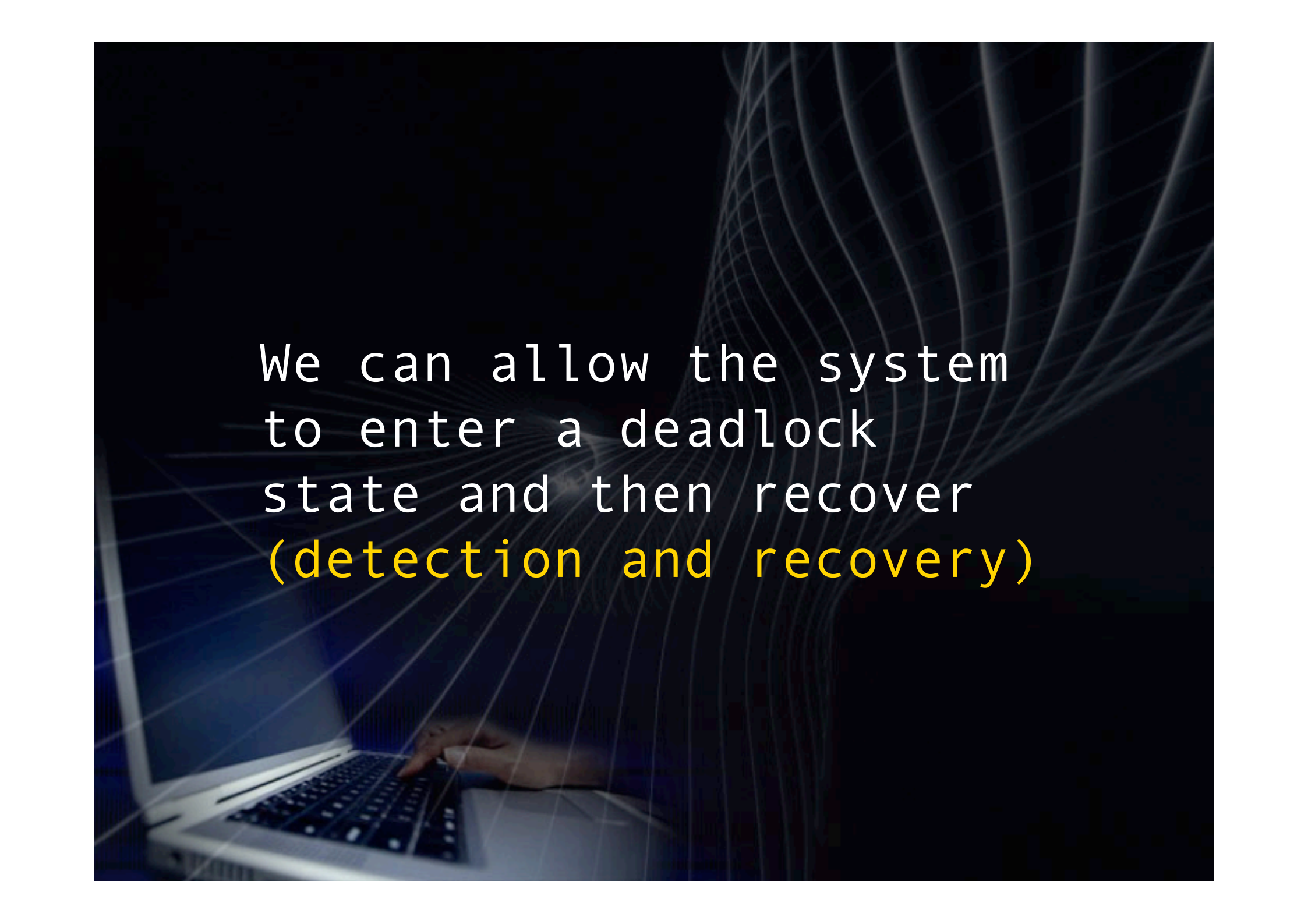
Each process involved in
the cycle is deadlocked
(a cycle in the graph is a
necessary and sufficient
condition for the
existence of a deadlock)

If each resource type has several instances, then **one cycle does not necessarily imply that a deadlock occurred** (the cycle is a necessary but not sufficient condition)



Methods for handling deadlock

We can use a protocol to ensure that the system will **never** enter a deadlock state (**deadlock prevention**)



We can allow the system
to enter a deadlock
state and then recover
(detection and recovery)

We can ignore the problem,
and pretend that deadlocks
never occur in the system

It is up to the
application developer
to write programs that
handle deadlocks

Deadlock prevention

Deadlock prevention is a set of methods for ensuring that at least one of the necessary conditions can never occur

mutual exclusion

It is not possible to prevent deadlocks by denying the mutual exclusion condition

hold-and-wait

That condition may be prevented by requiring that each process must release all the resources currently allocated before it can request any additional resources.

no preemption

If a process that it is holding same resources request another resource that cannot be immediately allocated to it, then all resources currently being held are preempted

circular wait

The condition can be prevented by defining a total ordering of all resource types and by requiring that each process requests resources in an increasing order

The background of the slide features a dark, moody aesthetic. On the right side, there is a close-up, slightly blurred image of a white pen resting on a lined notebook. The notebook's pages are visible, showing some faint, illegible handwriting. Overlaid on this scene are numerous thin, glowing white lines that curve and swirl across the frame, creating a sense of motion and depth. The overall color palette is dominated by dark blues, greys, and blacks, with the white text and pen providing high contrast.

We associate an index
with each resource type

Then R_i precedes R_j in
the ordering if $i < j$

Two processes A and B,
are deadlocked if A has
acquired R_i and requests
 R_j , and B has acquired R_j
and requests R_i

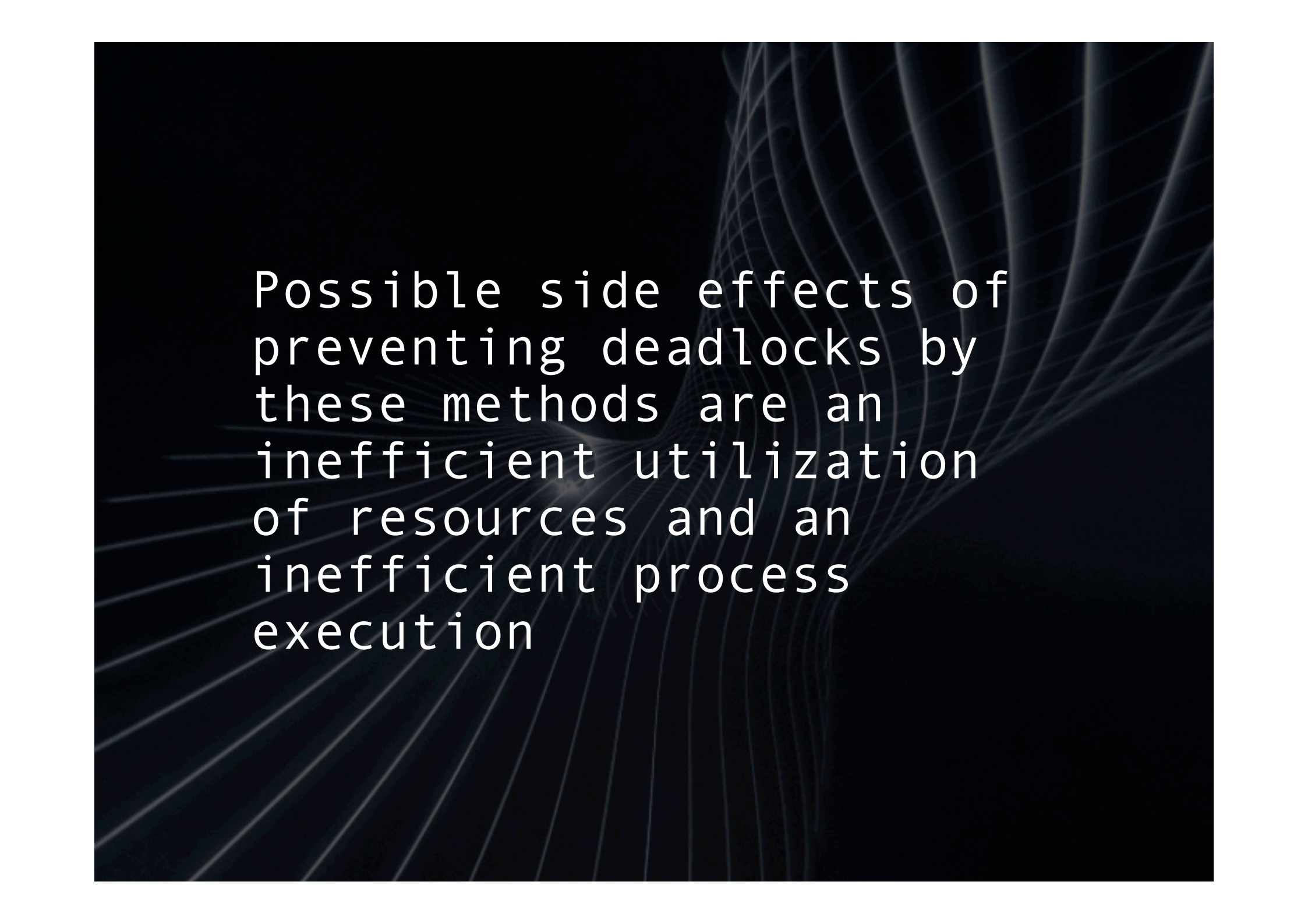
That condition is impossible because it implies $i < j$ and $j < i$

```
total memory usage: 4096k used, 4096k free, 4096k reserved, 4096k shared, 4096k private, 2628k wired, 1428k active, 2788k inactive, 18888k vtext, 128k free, 20911k pageins, 20948(0) pageouts

CPU  TIME  #TH  #PRCS  #PRVOT  #PMEM  #PWT  #PARD  #PICK  #VIRT
0.0%  0:00.00  1  26  71  304K  2728K  1928K  3438
0.0%  0:00.00  1  14  19  256K  624K  888K  748
0.0%  0:00.00  1  26  55  248K  248K  728K  728
0.0%  0:21.54  1  21  29  2832K  188K  2528K  748
0.0%  0:00.21  2  73  73  8956K  2368K  5148K  2448
0.0%  0:00.01  1  14  19  248K  624K  888K  748
0.0%  0:00.01  1  18  35  248K  268K  728K  728
0.0%  0:00.13  4  53  30  644K  2228K  1948K  878
0.0%  0:13.04  7  189  185  4248K  118  178  4198
0.0%  0:12.62  22  282  748  668  318  918  5078
0.1%  0:20.49  15  311  585  458  258  758  5188
0.1%  0:30.90  14  288  724  1438  188  1748  4278
0.1%  0:00.45  5  38  182  2908K  3128K  4148K  1808
0.1%  0:30.74  14  143  258  8956K  8848K  228  4488
0.1%  0:00.71  3  24  25  28K  272K  184K  748
0.1%  0:00.01  1  19  21  48K  184K  248K  748
0.1%  0:11.06  2  81  58  188K  148K  1288K  838
0.1%  0:00.23  2  25  28  68K  178K  272K  748
0.1%  2:29.29  28  352  919  1818  458  1858  13198
0.1%  0:00.14  1  58  83  184K  2278K  448K  3128
0.1%  0:00.09  1  49  42  124K  2112K  888K  2468
0.1%  0:01.38  2  99  81  1244K  404K  1288K  1188
0.1%  0:01.21  6  238  215  1272K  9452K  234K  488
```

Deadlock avoidance

Deadlock-prevention algorithms prevent deadlocks by constraining the strategy on how requests can be made



Possible side effects of preventing deadlocks by these methods are an inefficient utilization of resources and an inefficient process execution

With deadlock avoidance,
a decision is **made**
dynamically whether current
resource allocation
requests, if granted,
would potentially lead to
deadlock

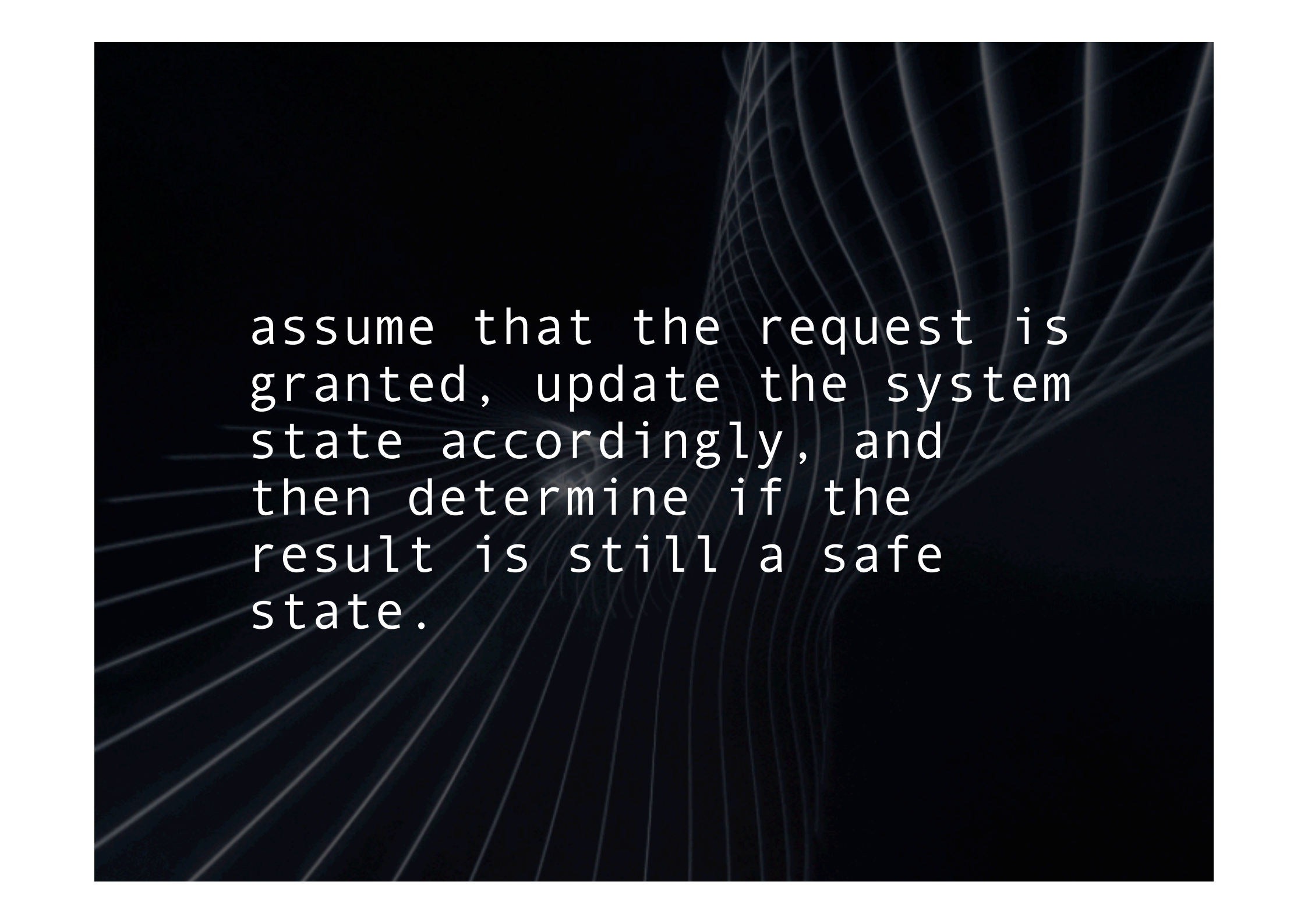
The resource allocation state is defined by the number of allocated and available resources and the maximum demands of processes

A **safe state** is one in which there is at least one process execution sequence such that all processes can be run to completion (**safe sequence**)

Banker's algorithm

When a process makes a request for a set of resources





assume that the request is granted, update the system state accordingly, and then determine if the result is still a safe state.

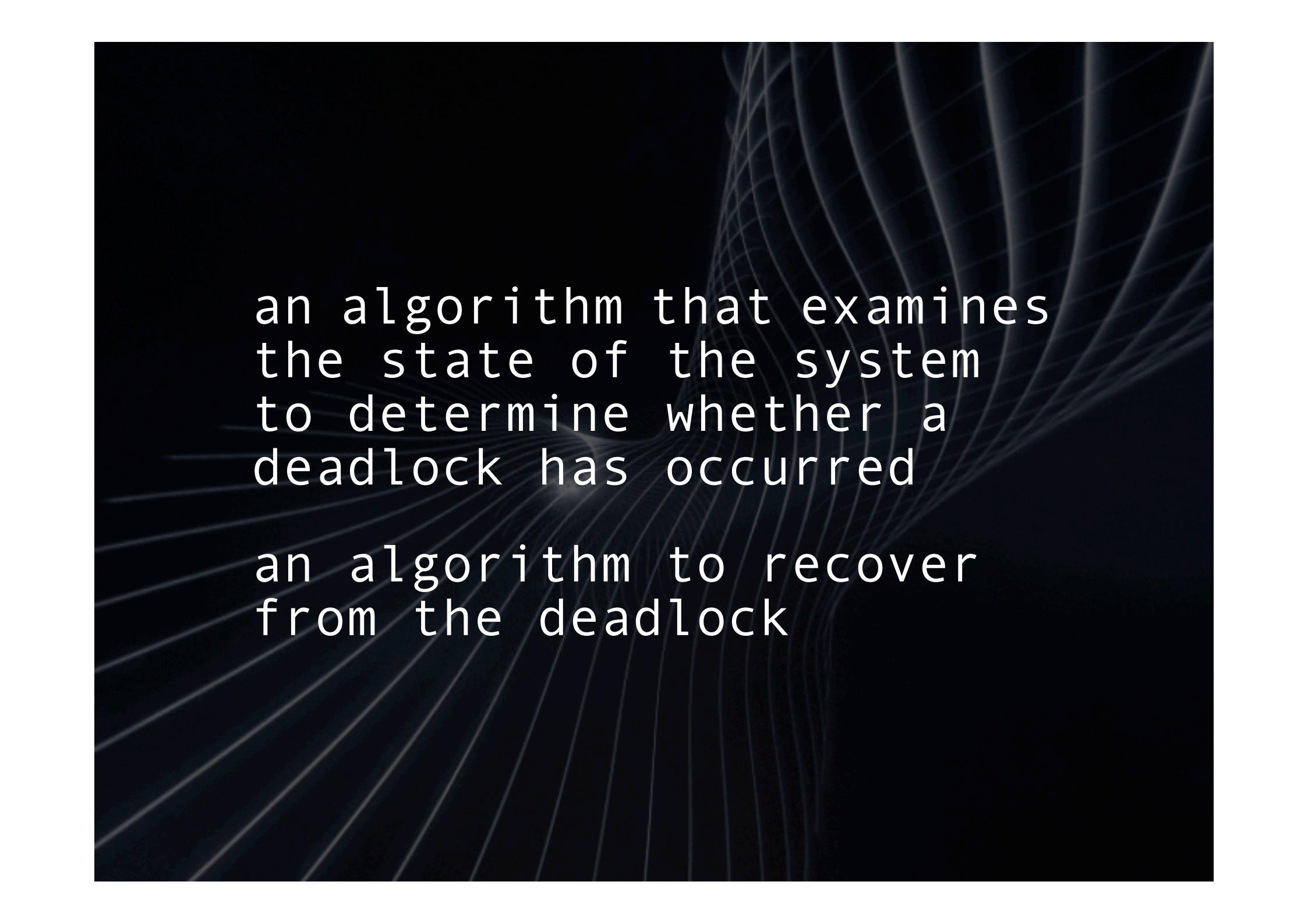


If so, grant the request,
if not, block the process
until it is safe
to grant the request

Deadlock detection

It requires:





an algorithm that examines
the state of the system
to determine whether a
deadlock has occurred

an algorithm to recover
from the deadlock

Recovery


Possible approaches:

- Abort all deadlocked processes



The image shows a blurred screenshot of a system monitor or command prompt window. The window displays a table of process information, likely from a Windows Task Manager or a similar utility. The table has several columns, including PID, Name, CPU, Private, Working Set, and Session ID. The data is presented in a grid format with alternating light and dark rows. The text is small and difficult to read due to the blur, but the structure is clearly that of a process list.

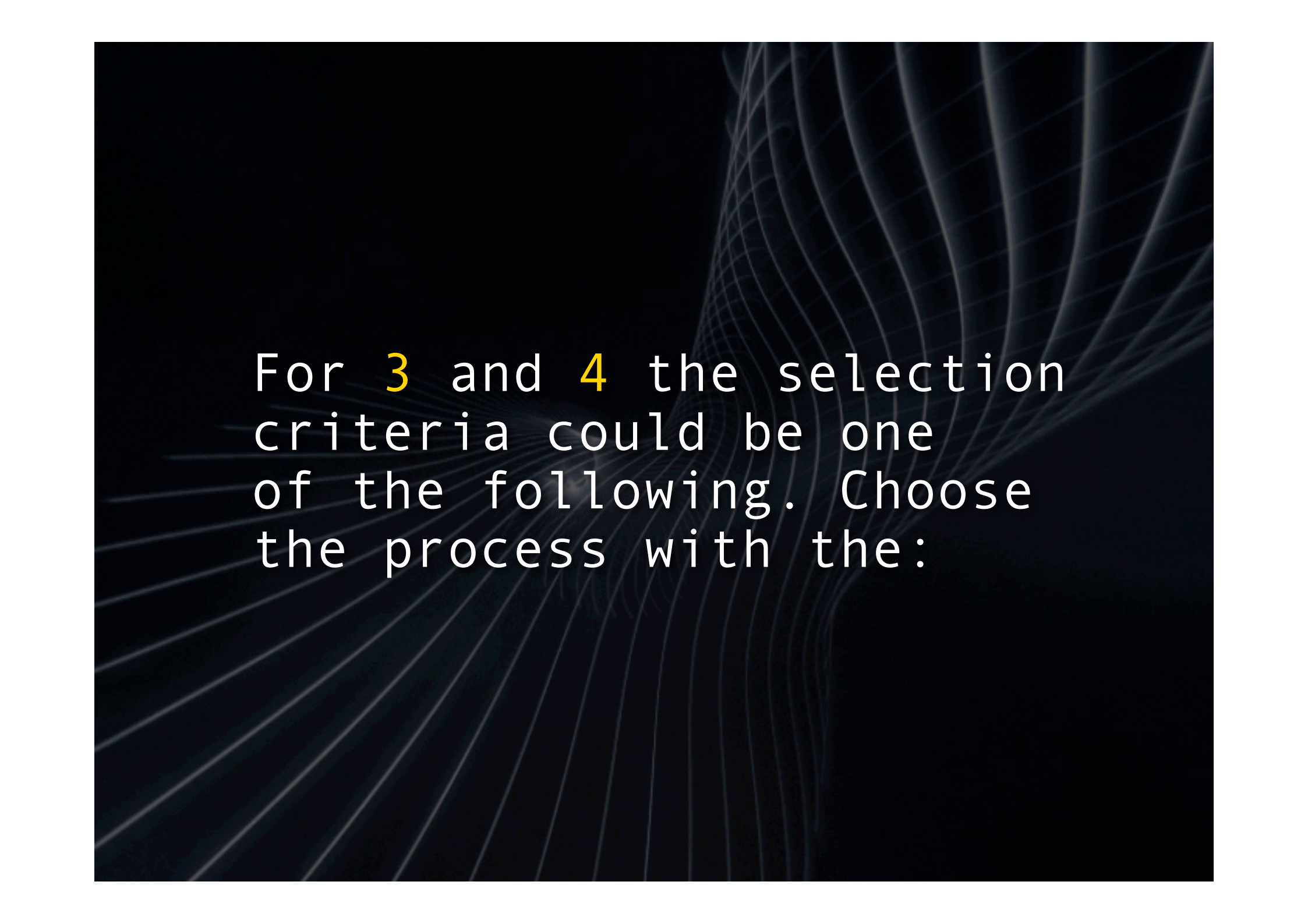
PID	Name	CPU	Private	Working Set	Session ID
1000	System	0%	4K	4K	0
4	smss.exe	0%	4K	4K	0
8	svchost.exe	0%	4K	4K	0
12	csrss.exe	0%	4K	4K	0
16	csrss.exe	0%	4K	4K	0
20	csrss.exe	0%	4K	4K	0
24	csrss.exe	0%	4K	4K	0
28	csrss.exe	0%	4K	4K	0
32	csrss.exe	0%	4K	4K	0
36	csrss.exe	0%	4K	4K	0
40	csrss.exe	0%	4K	4K	0
44	csrss.exe	0%	4K	4K	0
48	csrss.exe	0%	4K	4K	0
52	csrss.exe	0%	4K	4K	0
56	csrss.exe	0%	4K	4K	0
60	csrss.exe	0%	4K	4K	0
64	csrss.exe	0%	4K	4K	0
68	csrss.exe	0%	4K	4K	0
72	csrss.exe	0%	4K	4K	0
76	csrss.exe	0%	4K	4K	0
80	csrss.exe	0%	4K	4K	0
84	csrss.exe	0%	4K	4K	0
88	csrss.exe	0%	4K	4K	0
92	csrss.exe	0%	4K	4K	0
96	csrss.exe	0%	4K	4K	0
100	csrss.exe	0%	4K	4K	0
104	csrss.exe	0%	4K	4K	0
108	csrss.exe	0%	4K	4K	0
112	csrss.exe	0%	4K	4K	0
116	csrss.exe	0%	4K	4K	0
120	csrss.exe	0%	4K	4K	0
124	csrss.exe	0%	4K	4K	0
128	csrss.exe	0%	4K	4K	0
132	csrss.exe	0%	4K	4K	0
136	csrss.exe	0%	4K	4K	0
140	csrss.exe	0%	4K	4K	0
144	csrss.exe	0%	4K	4K	0
148	csrss.exe	0%	4K	4K	0
152	csrss.exe	0%	4K	4K	0
156	csrss.exe	0%	4K	4K	0
160	csrss.exe	0%	4K	4K	0
164	csrss.exe	0%	4K	4K	0
168	csrss.exe	0%	4K	4K	0
172	csrss.exe	0%	4K	4K	0
176	csrss.exe	0%	4K	4K	0
180	csrss.exe	0%	4K	4K	0
184	csrss.exe	0%	4K	4K	0
188	csrss.exe	0%	4K	4K	0
192	csrss.exe	0%	4K	4K	0
196	csrss.exe	0%	4K	4K	0
200	csrss.exe	0%	4K	4K	0
204	csrss.exe	0%	4K	4K	0
208	csrss.exe	0%	4K	4K	0
212	csrss.exe	0%	4K	4K	0
216	csrss.exe	0%	4K	4K	0
220	csrss.exe	0%	4K	4K	0
224	csrss.exe	0%	4K	4K	0
228	csrss.exe	0%	4K	4K	0
232	csrss.exe	0%	4K	4K	0
236	csrss.exe	0%	4K	4K	0
240	csrss.exe	0%	4K	4K	0
244	csrss.exe	0%	4K	4K	0
248	csrss.exe	0%	4K	4K	0
252	csrss.exe	0%	4K	4K	0
256	csrss.exe	0%	4K	4K	0
260	csrss.exe	0%	4K	4K	0
264	csrss.exe	0%	4K	4K	0
268	csrss.exe	0%	4K	4K	0
272	csrss.exe	0%	4K	4K	0
276	csrss.exe	0%	4K	4K	0
280	csrss.exe	0%	4K	4K	0
284	csrss.exe	0%	4K	4K	0
288	csrss.exe	0%	4K	4K	0
292	csrss.exe	0%	4K	4K	0
296	csrss.exe	0%	4K	4K	0
300	csrss.exe	0%	4K	4K	0
304	csrss.exe	0%	4K	4K	0
308	csrss.exe	0%	4K	4K	0
312	csrss.exe	0%	4K	4K	0
316	csrss.exe	0%	4K	4K	0
320	csrss.exe	0%	4K	4K	0
324	csrss.exe	0%	4K	4K	0
328	csrss.exe	0%	4K	4K	0
332	csrss.exe	0%	4K	4K	0
336	csrss.exe	0%	4K	4K	0
340	csrss.exe	0%	4K	4K	0
344	csrss.exe	0%	4K	4K	0
348	csrss.exe	0%	4K	4K	0
352	csrss.exe	0%	4K	4K	0
356	csrss.exe	0%	4K	4K	0
360	csrss.exe	0%	4K	4K	0
364	csrss.exe	0%	4K	4K	0
368	csrss.exe	0%	4K	4K	0
372	csrss.exe	0%	4K	4K	0
376	csrss.exe	0%	4K	4K	0
380	csrss.exe	0%	4K	4K	0
384	csrss.exe	0%	4K	4K	0
388	csrss.exe	0%	4K	4K	0
392	csrss.exe	0%	4K	4K	0
396	csrss.exe	0%	4K	4K	0
400	csrss.exe	0%	4K	4K	0
404	csrss.exe	0%	4K	4K	0
408	csrss.exe	0%	4K	4K	0
412	csrss.exe	0%	4K	4K	0
416	csrss.exe	0%	4K	4K	0
420	csrss.exe	0%	4K	4K	0
424	csrss.exe	0%	4K	4K	0
428	csrss.exe	0%	4K	4K	0
432	csrss.exe	0%	4K	4K	0
436	csrss.exe	0%	4K	4K	0
440	csrss.exe	0%	4K	4K	0
444	csrss.exe	0%	4K	4K	0
448	csrss.exe	0%	4K	4K	0
452	csrss.exe	0%	4K	4K	0
456	csrss.exe	0%	4K	4K	0
460	csrss.exe	0%	4K	4K	0
464	csrss.exe	0%	4K	4K	0
468	csrss.exe	0%	4K	4K	0
472	csrss.exe	0%	4K	4K	0
476	csrss.exe	0%	4K	4K	0
480	csrss.exe	0%	4K	4K	0
484	csrss.exe	0%	4K	4K	0
488	csrss.exe	0%	4K	4K	0
492	csrss.exe	0%	4K	4K	0
496	csrss.exe	0%	4K	4K	0
500	csrss.exe	0%	4K	4K	0
504	csrss.exe	0%	4K	4K	0
508	csrss.exe	0%	4K	4K	0
512	csrss.exe	0%	4K	4K	0
516	csrss.exe	0%	4K	4K	0
520	csrss.exe	0%	4K	4K	0
524	csrss.exe	0%	4K	4K	0
528	csrss.exe	0%	4K	4K	0
532	csrss.exe	0%	4K	4K	0
536	csrss.exe	0%	4K	4K	0
540	csrss.exe	0%	4K	4K	0
544	csrss.exe	0%	4K	4K	0
548	csrss.exe	0%	4K	4K	0
552	csrss.exe	0%	4K	4K	0
556	csrss.exe	0%	4K	4K	0
560	csrss.exe	0%	4K	4K	0
564	csrss.exe	0%	4K	4K	0
568	csrss.exe	0%	4K	4K	0
572	csrss.exe	0%	4K	4K	0
576	csrss.exe	0%	4K	4K	0
580	csrss.exe	0%	4K	4K	0
584	csrss.exe	0%	4K	4K	0
588	csrss.exe	0%	4K	4K	0
592	csrss.exe	0%	4K	4K	0
596	csrss.exe	0%	4K	4K	0
600	csrss.exe	0%	4K	4K	0
604	csrss.exe	0%	4K	4K	0
608	csrss.exe	0%	4K	4K	0
612	csrss.exe	0%	4K	4K	0
616	csrss.exe	0%	4K	4K	0
620	csrss.exe	0%	4K	4K	0
624	csrss.exe	0%	4K	4K	0
628	csrss.exe	0%	4K	4K	0
632	csrss.exe	0%	4K	4K	0
636	csrss.exe	0%	4K	4K	0
640	csrss.exe	0%	4K	4K	0
644	csrss.exe	0%	4K	4K	0
648	csrss.exe	0%	4K	4K	0
652	csrss.exe	0%	4K	4K	0
656	csrss.exe	0%	4K	4K	0
660	csrss.exe	0%	4K	4K	0
664	csrss.exe	0%	4K	4K	0
668	csrss.exe	0%	4K	4K	0
672	csrss.exe	0%	4K	4K	0
676	csrss.exe	0%	4K	4K	0
680	csrss.exe	0%	4K	4K	0
684	csrss.exe	0%	4K	4K	0
688	csrss.exe	0%	4K	4K	0
692	csrss.exe	0%	4K	4K	0
696	csrss.exe	0%	4K	4K	0
700	csrss.exe	0%	4K	4K	0
704	csrss.exe	0%	4K	4K	0
708	csrss.exe	0%	4K	4K	0
712	csrss.exe	0%	4K	4K	0
716	csrss.exe	0%	4K	4K	0
720	csrss.exe	0%	4K	4K	0
724	csrss.exe	0%	4K	4K	0
728	csrss.exe	0%	4K	4K	0
732	csrss.exe	0%	4K	4K	0
736	csrss.exe	0%	4K	4K	0
740	csrss.exe	0%	4K	4K	0
744	csrss.exe	0%	4K	4K	0
748	csrss.exe	0%	4K	4K	0
752	csrss.exe	0%	4K	4K	0
756	csrss.exe	0%	4K	4K	0
760	csrss.exe	0%	4K	4K	0
764	csrss.exe	0%	4K	4K	0
768	csrss.exe	0%	4K	4K	0
772	csrss.exe	0%	4K	4K	0
776	csrss.exe	0%	4K	4K	0
780	csrss.exe	0%	4K	4K	0
784	csrss.exe	0%	4K	4K	0
788	csrss.exe	0%	4K	4K	0
792	csrss.exe	0%	4K	4K	0
796	csrss.exe	0%	4K	4K	0
800	csrss.exe	0%	4K	4K	0
804	csrss.exe	0%	4K	4K	0
808	csrss.exe	0%	4K	4K	0
812	csrss.exe	0%	4K	4K	0
816	csrss.exe	0%	4K	4K	0
820	csrss.exe	0%	4K	4K	0
824	csrss.exe	0%	4K	4K	0
828	csrss.exe	0%	4K	4K	0
832	csrss.exe	0%	4K	4K	0
836	csrss.exe	0%	4K	4K	0
840	csrss.exe	0%	4K	4K	0
844	csrss.exe	0%	4K	4K	0
848	csrss.exe	0%	4K	4K	0
852	csrss.exe	0%	4K	4K	0
856	csrss.exe	0%	4K	4K	0
860	csrss.exe	0%	4K	4K	0
864	csrss.exe	0%	4K	4K	0
868	csrss.exe	0%	4K	4K	0
872	csrss.exe	0%	4K	4K	0
876	csrss.exe	0%	4K	4K	0
880	csrss.exe	0%	4K	4K	0
884	csrss.exe	0%	4K	4K	0
888	csrss.exe	0%	4K	4K	0
892	csrss.exe	0%	4K	4K	0
896	csrss.exe	0%	4K	4K	0
900	csrss.exe	0%	4K	4K	0
904	csrss.exe	0%	4K	4K	0
908	csrss.exe	0%	4K	4K	0
912	csrss.exe	0%	4K	4K	0
916	csrss.exe	0%	4K	4K	0
920	csrss.exe	0%	4K	4K	0
924	csrss.exe	0%	4K	4K	0
928	csrss.exe	0%	4K	4K	0
932	csrss.exe	0%	4K	4K	0
936	csrss.exe	0%	4K	4K	0
940	csrss.exe	0%	4K	4K	0
944	csrss.exe	0%	4K	4K	0
948	csrss.exe	0%	4K	4K	0
952	csrss.exe	0%	4K	4K	0
956	csrss.exe	0%	4K	4K	0
960	csrss.exe	0%	4K	4K	0
964	csrss.exe	0%	4K	4K	0
968	csrss.exe	0%	4K	4K	0
972	csrss.exe	0%	4K	4K	0
976	csrss.exe	0%	4K	4K	0
980	csrss.exe	0%	4K	4K	0
984	csrss.exe	0%	4K	4K	0
988	csrss.exe	0%	4K	4K	0
992	csrss.exe	0%	4K	4K	0
996	csrss.exe	0%	4K	4K	0
1000	csrss.exe	0%	4K	4K	0



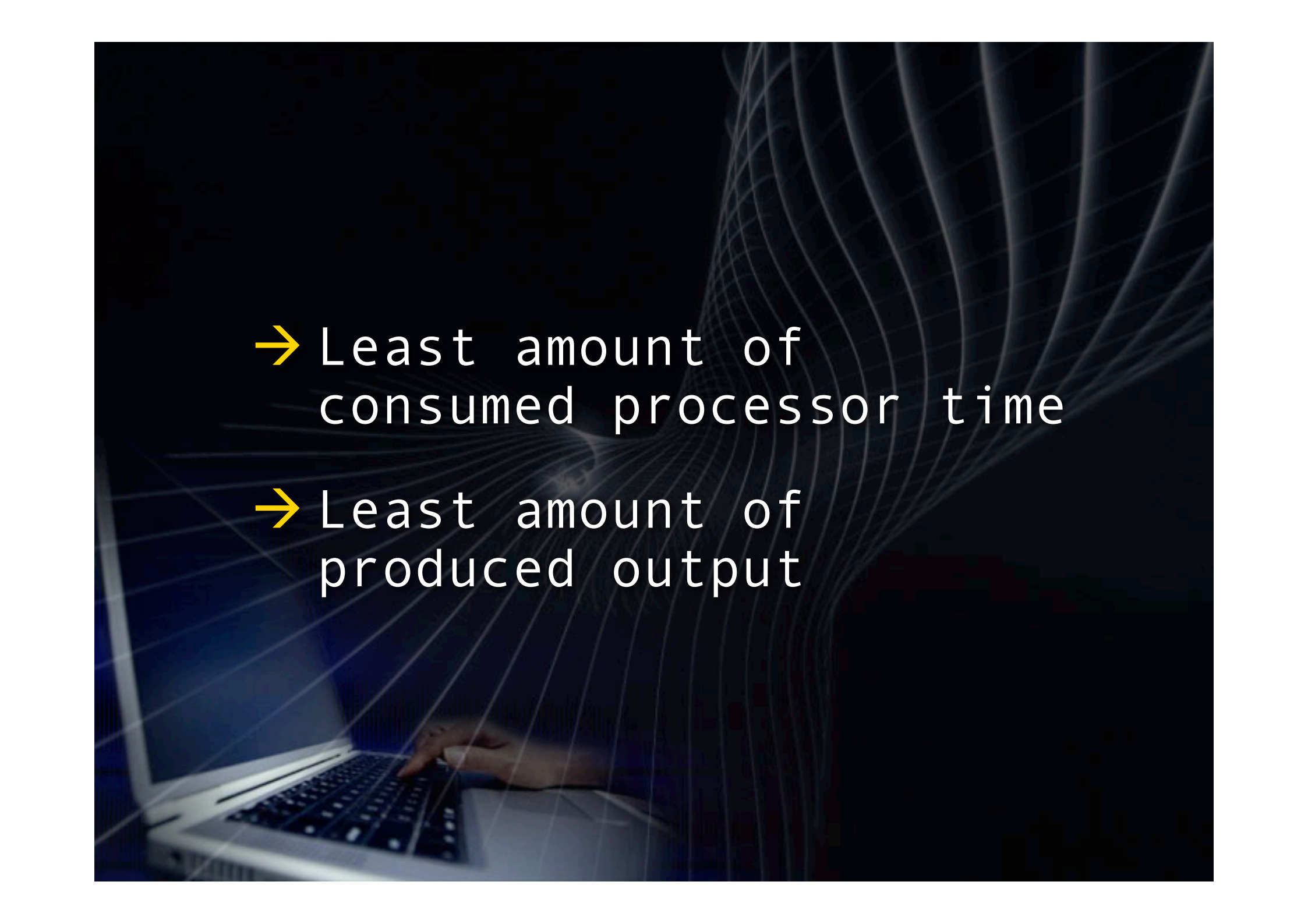
Back up each deadlocked process to some previously defined checkpoints, and restart all processes from those checkpoints


3. Successively abort
deadlocked processes
until deadlock not
longer exists

4. Successively preempt
resources until deadlock
not longer exists



For 3 and 4 the selection criteria could be one of the following. Choose the process with the:

- 
- Least amount of consumed processor time
 - Least amount of produced output

- 
- Most estimated remaining time
 - Least total resources allocated so far
 - Lowest priority