Running.c

```
int main()
{
do { } while (1);
}
```

```
(base) 202020827@cslinux:~/os/block$ gcc -o running running.c
(base) 202020827@cslinux:~/os/block$ ./running &
[1] 1023986
(base) 202020827@cslinux:~/os/block$ ps -l
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1096 1020912 1011711  0  80   0 -  3218 do_wai pts/70   00:00:00 bash
0 R  1096 1023986 1020912 99  80   0 -   661 -      pts/70   00:00:02 running
0 R  1096 1024002 1020912  0  80   0 -  3496 -      pts/70   00:00:00 ps
(base) 202020827@cslinux:~/os/block$
```

1. ps -l 명령을 수행하면 프로세스의 S 항목(STATE)에 R(Running)이 표시된다.
2. ps -l 명령을 여러 번 반복해도 상태 표시가 대체로 R 로 유지된다.프로세스가 CPU 를 계속 사용하려 하므로, 사용자 입장에서는 계속 Running 으로 보인다.
3. 표시된 R 상태는 프로세스가 실행 가능한 상태(실행 중이거나 곧 실행될 수 있는 상태)를 의미한다.
4. 실제로는 CPU 스케줄링에 따라 프로세스가 미세하게 실행과 중단을 반복하지만, 블로킹되지 않는 이상 사용자 명령(ps)에서 확인할 때는 계속 R 상태로 보이게 된다.

Blocked1.c

```
#include <unistd.h>

int main(){
   sleep(3600);
}
```

```
(base) 202020827@cslinux:~/os/block$ gcc -o blocked1 blocked1.c
(base) 202020827@cslinux:~/os/block$ ./blocked1 &
[1] 1019432
(base) 202020827@cslinux:~/os/block$ ps -l
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1096 1013757 1011711  0  80   0 -  3218 do_wai pts/76   00:00:00 bash
0 S  1096 1019432 1013757  0  80   0 -   661 hrtime pts/76   00:00:00 blocked1
0 R  1096 1019496 1013757  0  80   0 -  3496 -      pts/76   00:00:00 ps
```

1. ps -l 명령을 수행하면 프로세스의 S 항목(STATE)에 S(Sleeping)가 표시된다.
2. ps -l 명령을 여러 번 반복해도 상태 표시가 대체로 S 로 유지된다. sleep 함수로 인해 프로세스가 계속 대기 상태에 놓여있으므로, 사용자 입장에서는 계속 Sleeping 으로 보인다.

3. 표시된 S 상태는 프로세스가 인터럽트 가능한 대기 상태에 있음을 의미한다.
4. 실제로는 지정된 시간(3600 초) 동안 CPU 를 사용하지 않고 대기하고 있으므로, 프로세스의 상태는 계속 변하지 않고 같은 상태로 유지된다.

Blocked2.c

```
#include <stdio.h>

int main(){
    int a;
    scanf("%d", &a);
}
(base) 202020827@cslinux:~/os/block$ ps -l -t pts/1
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1000    4637    4600  0  80   0 -  3177 -      pts/1    00:00:00 bash
(base) 202020827@cslinux:~/os/block$ ps -ef | grep blocked2
2023209+ 1021248   60962  0 15:33 pts/23   00:00:00 ./blocked2
2023209+ 1021667 1015969  0 15:34 pts/91   00:00:00 ./blocked2
2020213+ 1022860 1012480  0 15:36 pts/61   00:00:00 ./blocked2
2021210+ 1024226 1013793  0 15:38 pts/77   00:00:00 ./blocked2
2021210+ 1025004 1014734  0 15:41 pts/93   00:00:00 ./blocked2 foreground
2020209+ 1025483 1012586  0 15:43 pts/66   00:00:00 vi blocked2.c
2023210+ 1025816    5130  0 15:44 pts/4    00:00:00 ./blocked2
jhkim    1028332 1027470  0 15:48 pts/125  00:00:00 ./blocked2
2020208+ 1029125 1020912  0 15:49 pts/70   00:00:00 ./blocked2
2020214+ 1029828 1029400  0 15:50 pts/73   00:00:00 vi blocked2.c
2020203+ 1032010 1018067  0 15:53 pts/104  00:00:00 ./blocked2
2021210+ 1032768 1015573  0 15:55 pts/85   00:00:00 ./blocked2
2022218+ 1033578 1015966  0 15:57 pts/89   00:00:00 ./blocked2
2023210+ 1033677 1030198  0 15:57 pts/130  00:00:00 ./blocked2
2020208+ 1034331 1020983  0 16:00 pts/76   00:00:00 grep --color=auto blocked2
```

1. 프로세스의 상태
    - scanf 함수로 인해 표준 입력(stdin)에서 데이터를 기다리는 중이다.
    - ps 로 확인했을 때 주로 S(Sleeping) 상태로 표시된다.
2. 실제 상태 변화 여부
    - 사용자가 입력을 하지 않는 한, 프로세스는 계속 대기 상태(인터럽트 가능한 대기 상태)로 머무른다.
    - 내부적으로는 OS 스케줄러에 의해 잠깐씩 깨워지거나 다시 대기하는 과정을 거치지만, 사용자 입장에서는 대체로 변하지 않고 계속 S 상태로 보이게 된다.
    - 사용자가 값을 입력하면, 잠시 Running 상태가 되었다가 입력 처리를 완료한 뒤 종료(또는 다음 로직 수행)하게 된다.

Blocked3.c

```c
#include <unistd.h>

int main() {
    int i;
    do {
        sleep(1);
        for(i = 0; i < 100000000; i++)
        ;
    } while(1);
}
```

```
(base) 202020827@cslinux:~/os/block$ gcc -o blocked3 blocked3.c
(base) 202020827@cslinux:~/os/block$ ./blocked3 &
[1] 1023016
(base) 202020827@cslinux:~/os/block$ ps -l
F S   UID     PID    PPID  C PRI  NI ADDR SZ WCHAN  TTY          TIME CMD
0 S  1096 1020912 1011711  0  80   0 -  3218 do_wai pts/70   00:00:00 bash
0 S  1096 1023016 1020912 15  80   0 -   661 hrtime pts/70   00:00:00 blocked3
0 R  1096 1023032 1020912  0  80   0 -  3496 -      pts/70   00:00:00 ps
```

1. ps -l 명령을 여러 번 수행하면 시점에 따라 S(Sleeping) 또는 R(Running)으로 번갈아 표시된다.

2. sleep(1) 이후 CPU 를 한동안 점유하는 과정을 반복하므로, 실제로 프로세스는 대기 상태와 실행 상태를 주기적으로 오가고 있는다.

3. 따라서 ps -l 결과에서도 프로세스의 상태가 S 와 R 로 계속 바뀌어 나타난다.