ccOS Lifecycle

Infotainment Software Development Team

Hyundai Motor Company

Contents

1 서론

1.1 목적

본 문서는 ccOS의 Lifecycle을 설명한다. Lifecycle은 개체가 생성(creation)하고 소멸(termination)하는 동안의 상태 변화를 정형화된 형태로 기술한다. 개체의 상태 변화는

ccOS는 Linux를 기본으로 하고 있기 때문에, Linux의 lifecycle을 리뷰한다.

2 Linux Process Lifecycle

들어가기 앞서, Linux Process Lifecycle은 Linux Tutorial의 내용에서 참고하였다.

From the time a process is created with a fork() until it has completed its job and disappears from the process table, it goes through many different states. The state a process is in changes many times during its “life.” These changes can occur, for example, when the process makes a system call, it is someone else’s turn to run, an interrupt occurs, or the process asks for a resource that is currently not available.

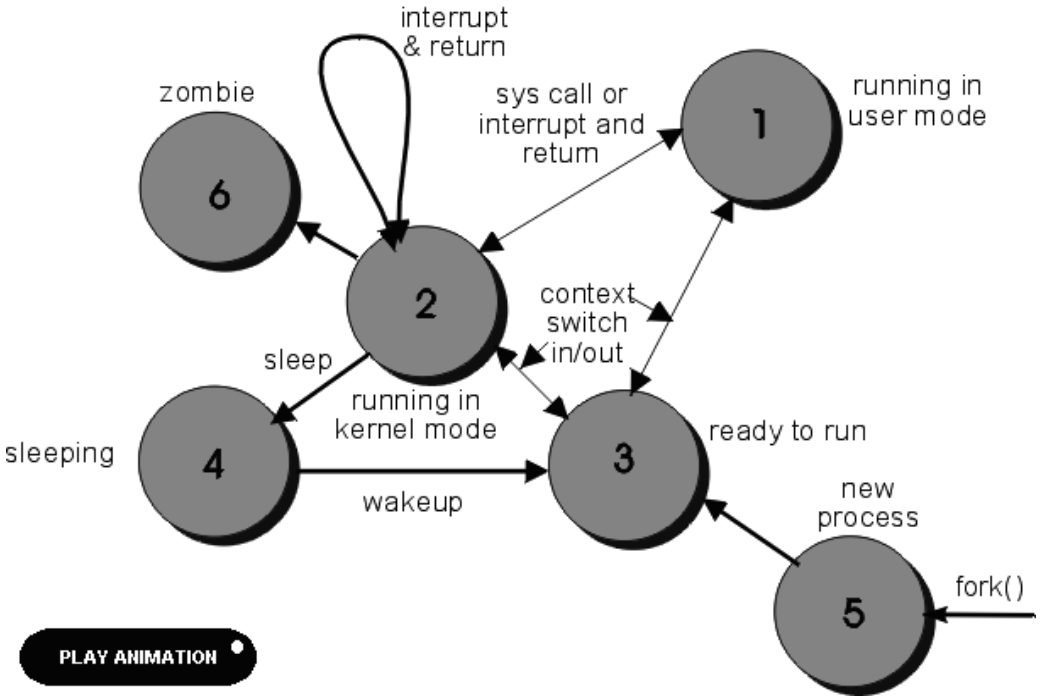

FIGURE 1: THE SKY IS THE LIMIT.

A commonly used model shows processes operating in one of six separate states, which you can find in sched.h:

executing in user mode executing in kernel mode ready to run sleeping newly created, not ready to run, and not sleeping issued exit system call (zombie)

The states listed here describe what is happening conceptually and do not indicate what ”official” state a process is in. The official states are listed below:

TASK_RUNNING task (process) currently running TASK_INTERRUPTABLE process is sleeping but can be woken up (interrupted) TASK_UNINTERRUPTABLE process is sleeping but can not be woken up (interrupted) TASK_ZOMBIE process terminated but its status was not collected (it was not waited for) TASK_STOPPED process stopped by a debugger or job control TASK_SWAPPING (removed in 2.3.x kernel)

Table - Process States in sched.h

In my list of states, there was no mention of a process actually being on the processor (TASK_RUNNING). Processes that are running in kernel mode or in user mode are both

in the TASK_RUNNING state. Although there is no 1:1 match-up, I hope you'll see what each state means as we go through the following description. You can see how this all looks graphically in the figure below.

3 Logical View