1. 소스코드

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <unistd.h>  #include <sys/wait.h>  void ssu\_echo\_exit(int status);  int main(void) {  pid\_t pid;  int status;  if((pid = fork()) < 0) {  fprintf(stderr, "fork error\n");  exit(1);  }  else if(pid == 0)//자식 프로세스 실행  exit(7);  if(wait(&status) != pid) {//부모 프로세스가 아닐 경우  fprintf(stderr, "wait error\n");  exit(1);  }  ssu\_echo\_exit(status);  if((pid = fork()) < 0) {  fprintf(stderr, "fork error\n");  exit(1);  }  else if(pid == 0)//자식 프로세스 실행  abort();//프로세스가 시그널을 받아 비정상적 종료  if(wait(&status) != pid) {//부모 프로세스가 아닐 경우  fprintf(stderr, "wait error\n");  exit(1);  }  ssu\_echo\_exit(status);  if((pid = fork()) < 0) {  fprintf(stderr, "fork error\n");  exit(1);  }  else if(pid == 0)//자식 프로세스 실행  status /= 0;//프로세스가 시그널을 받아 비정상적 종료  if(wait(&status) != pid) {//부모 프로세스가 아닐 경우  fprintf(stderr, "wait error\n");  exit(1);  }  ssu\_echo\_exit(status);  exit(0);  }  void ssu\_echo\_exit(int status) {  if(WIFEXITED(status))//자식 프로세스가 정상적으로 종료된 경우  printf("normal termination, exit status = %d\n", WEXITSTATUS(status));  else if(WIFSIGNALED(status))//자식 프로세스가 시그널을 받았으나 시그널을 처리하지 않아 비정상적으로 종료된 경우  printf("abnormal termination, signal number = %d%s\n", WTERMSIG(status),//시그널 번호 반환  //코어 파일이 생성된 경우  #ifdef WCOREDUMP  WCOREDUMP(status) ? " (core file generated)" : "");  #else  "");  #endif  //자식 프로세스가 현재 중지상태이면  else if(WIFSTOPPED(status))  printf("child stopped, signal number = %d\n", WSTOPSIG(status));//실행을 일시 중단시킨 시그널 번호 반환  } |

2. 실행결과

