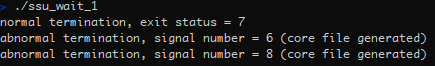
컴퓨터학부 20162448 김병준

1. 결과



1. 소스코드

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <unistd.h>  #include <sys/types.h>  #include <sys/wait.h>  void ssu\_echo\_exit(int stauts);  int main(void)  {  pid\_t pid;  int status;  if((pid = fork()) < 0) { // 1. 자식 프로세스 호출  fprintf(stderr, "fork error\n");  exit(1);  }  else if(pid == 0) // 2. 자식 프로세스는 상태 7 반환 및 종료  exit(7);  if(wait(&status) != pid) { // 3. 부모 프로세스는 자식 프로세스 상태를 status에 저장, 실행중:0. 종료:pid,  fprintf(stderr, "wait error\n");  exit(1);  }  ssu\_echo\_exit(status); // 4. 상태 출력  if((pid = fork()) < 0) { // 5. 자식 호출  fprintf(stderr, "fork error\n");  exit(1);  }  else if(pid == 0) // 6. 자식 비정상 종료  abort();  if(wait(&status) != pid) { // 7. 자식 상태 저장 및 정상 종료가 아닐 경우  fprintf(stderr, "wait error");  exit(1);  }  ssu\_echo\_exit(status); // 8. 상태 출력  if((pid = fork()) < 0) { // 9. 자식 호출  fprintf(stderr, "fork error\n");  exit(1);  }  else if(pid == 0) // 10. 자식프로세스는 상태값 0으로 나눔  status /= 0;  if(wait(&status) != pid) { // 11. 자식 상태 저장 및 정상종료가 아닐 경우  fprintf(stderr, "wait error");  exit(1);  }  ssu\_echo\_exit(status); // 상태 출력  exit(0);  }  void ssu\_echo\_exit(int status) {  if(WIFEXITED(status)) // 자식 프로세스 정상 종료  printf("normal termination, exit status = %d\n", WEXITSTATUS(status)); // 반환값 출력  else if(WIFSIGNALED(status)) // 시그널은 받고 비정상 종료  printf("abnormal termination, signal number = %d%s\n", WTERMSIG(status), // 시그널값 출력  #ifdef WCOREDUMP // 코어파일이 생성된 경우  WCOREDUMP(status) ? " (core file generated)" : "");  #else  "");  #endif  else if(WIFSTOPPED(status)) // 중지상태일 경우  printf("child stopped, signal number = %d\n", WSTOPSIG(status)); // 중지 시그널 출력  } |