1. 소스코드

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <unistd.h>  #include <errno.h>  #include <setjmp.h>  #include <signal.h>  #include <time.h>  static void ssu\_alarm(int signo);  static void ssu\_func(int signo);  void ssu\_mask(const char \*str);  static volatile sig\_atomic\_t can\_jump;  static sigjmp\_buf jump\_buf;  int main(void) {  //SIGUSR1 시그널에 대해 ssu\_func 시그널 핸들러 함수 등록  if(signal(SIGUSR1, ssu\_func) == SIG\_ERR) {  fprintf(stderr, "SIGUSR1 error");  exit(1);  }  //SIGALRM 시그널에 대해 ssu\_alarm 시그널 핸들러 함수 등록  if(signal(SIGALRM, ssu\_alarm) == SIG\_ERR) {  fprintf(stderr, "SIGALRM error");  exit(1);  }  ssu\_mask("starting main: ");  //siglongjmp() 호출에서 반환된 경우  //시그널 마스크를 복원하여 jump\_buf에 저장  if(sigsetjmp(jump\_buf, 1)) {  ssu\_mask("ending main: ");  exit(0);  }  can\_jump = 1;  while(1)  pause();  exit(0);  }  void ssu\_mask(const char \*str) {  sigset\_t sig\_set;  int err\_num;  err\_num = errno;  //시그널 마스크가 변하지 않음  if(sigprocmask(0, NULL, &sig\_set) < 0) {  printf("sigprocmask() error");  exit(1);  }  printf("%s", str);  //sig\_set에 존재하는 시그널 찾기  if(sigismember(&sig\_set, SIGINT))  printf("SIGINT ");  if(sigismember(&sig\_set, SIGINT))  printf("SIGQUIT ");  if(sigismember(&sig\_set, SIGUSR1))  printf("SIGUSR1 ");  if(sigismember(&sig\_set, SIGALRM))  printf("SIGALRM ");  printf("\n");  errno = err\_num;  }  //SIGUSR1의 시그널 핸들러 함수  static void ssu\_func(int signo) {  time\_t start\_time;  if(can\_jump == 0)  return;  ssu\_mask("starting ssu\_func: ");  alarm(3);  start\_time = time(NULL);    while(1)  if(time(NULL) > start\_time + 5)  break;  ssu\_mask("ending ssu\_func: ");  can\_jump = 0;  //sigsetjmp() 호출  siglongjmp(jump\_buf, 1);  }  //SIGALRM의 시그널 핸들러 함수  static void ssu\_alarm(int signo) {  ssu\_mask("in ssu\_alarm: ");  } |

2. 실행결과

