1. 소스코드

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <errno.h>  #include <signal.h>  static void ssu\_func(int signo);  void ssu\_print\_mask(const char \*str);  int main(void) {  sigset\_t new\_mask, old\_mask, wait\_mask;  ssu\_print\_mask("program start: ");  //SIGINT 시그널에 대한 시그널 핸들러 함수로 ssu\_func 등록  if(signal(SIGINT, ssu\_func) == SIG\_ERR) {  fprintf(stderr, "signal(SIGINT) error\n");  exit(1);  }  //wait\_mask를 0으로 초기화  sigemptyset(&wait\_mask);  //wait\_mask에 SIGUSR1 시그널을 추가  sigaddset(&wait\_mask, SIGUSR1);  //new\_mask를 0으로 초기화  sigemptyset(&new\_mask);  //new\_mask에 SIGINT 시그널을 추가  sigaddset(&new\_mask, SIGINT);    //new\_mask를 블록설정, old\_mask에 기존의 시그널 집합 저장  if(sigprocmask(SIG\_BLOCK, &new\_mask, &old\_mask) < 0) {  fprintf(stderr, "SIG\_BLOCK() error\n");  exit(1);  }  ssu\_print\_mask("in critical region: ");  //wait\_mask로 마스크된 시그널들이 발생할 때까지 일시정지  if(sigsuspend(&wait\_mask) != -1) {  fprintf(stderr, "sigsuspend() error\n");  exit(1);  }  ssu\_print\_mask("after return from sigsuspend: ");  //old\_mask로 마스크  if(sigprocmask(SIG\_SETMASK, &old\_mask, NULL) < 0) {  fprintf(stderr, "SIG\_SETMASK() error\n");  exit(1);  }  ssu\_print\_mask("program exit: ");  exit(0);  }  void ssu\_print\_mask(const char \*str) {  sigset\_t sig\_set;  int err\_num;  err\_num = errno;  //시그널 마스크가 안변함  if(sigprocmask(0, NULL, &sig\_set) < 0) {  fprintf(stderr, "sigprocmask() error\n");  exit(1);  }  printf("%s", str);  //sig\_set에 존재하는 시그널 찾기  if(sigismember(&sig\_set, SIGINT))  printf("SIGINT ");  if(sigismember(&sig\_set, SIGQUIT))  printf("SIGQUIT ");  if(sigismember(&sig\_set, SIGUSR1))  printf("SIGUSR1 ");  if(sigismember(&sig\_set, SIGALRM))  printf("SIGALRM ");  printf("\n");  errno = err\_num;  }  //SIGINT의 시그널 핸들러 함수  static void ssu\_func(int signo) {  ssu\_print\_mask("\nin ssu\_func: ");  } |

2. 실행결과

