결함 허용 프로세스 복구 기법

김신호*, 임성락**
*호서대학교 벤처전문대학원 컴퓨터응용기술
**호서대학교 컴퓨터공학부
e-mail:jerry_builder@hotmail.com

A recovery scheme of the Fault-tolerant Process

Sin-Ho Kim*, Seong-Rak Rim**
*Application of Computer Technology,
Graduate School of Venture, Hoseo University
**Dept of Computer Engineering, Hoseo University

요 약

프로세스 복구 기법은 장시간 실행을 요하는 프로세스에서 시스템 결함으로 인하여 발생하는 심각한 피해를 최소화하기 위하여 절대적으로 요구된다. 프로세스의 무결성을 지원하기 위한 예방 및 회피책은 결함 발생의 원인 규명과 예측에 의한 오버헤드가 있다. 본 논문에서는 이러한 오버헤드를 최소화하기 위한 결함 허용 프로세스 복구 기법을 제시한다. 제시한 기법은 프로세스의 실행 상태 저장 및 복구 기능을 위한 두 개의 시스템 호출을 설계하고, 리눅스 커널 2.4.18 내부에 구현하여 그 타당성을 검토하였다.

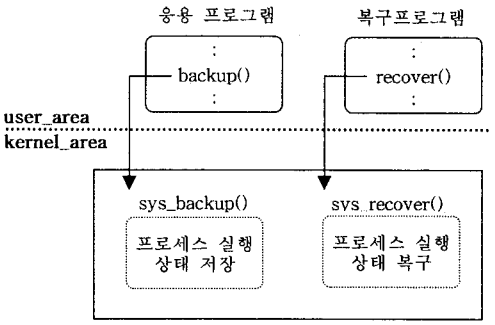
1. 서론

결함 허용 프로세스 복구 기법은 예상치 못한 결함 발생으로 인한 피해를 최소화하는 소프트웨어적인 도구이다[1]. 이 기법은 실행되고 있는 프로세스의 상태를 주기적으로 저장하여 시스템의 오류로 인해 예기치 못한 결함이 발생하였을 때, 결함이 발생하기 전까지의 프로세스 실행 상태를 복구하여 프로세스가 계속적으로 실행될 수 있도록 한다. 프로세스의 실행 상태 저장이란, 정상적으로 실행되고 있는 프로세스의 상태 정보를 저장하는 일을 말한다.

프로세스의 무결성을 보장하기 위한 예방 및 회피책은 결함 발생의 원인 규명과 예측에 의한 오버헤드가 있다[2]. 따라서 본 논문에서는 이러한 오버헤드를 최소화하기 위하여 결함 허용 프로세스 복구 기법을 제시하고, 사용자의 편의성과 시스템의 효율성을 제공하기 위하여 프로세스의 실행 상태 저장 및 복구 기능을 커널 내부에 구현하였다.

2. 설계

본 논문에서는 결함 허용 프로세스의 복구를 위하여 (그림 1)과 같이 두 개의 시스템 호출(backup, recover)을 추가한다.

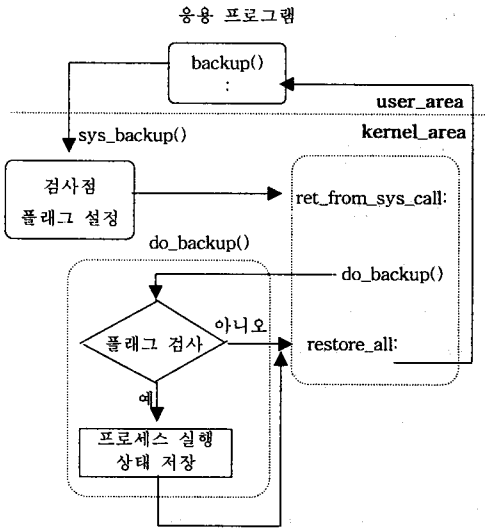


(그림 1) 시스템 호출 추가

(그림 1)에서 응용 프로그램은 프로세스의 실행 상태를 저장하기 위하여 backup()을 호출하고, 복구 프로그램에서는 프로세스의 실행 상태를 복구하기 위하여 recover()를 호출한다.

2.1 sys_backup() 함수

프로세스의 실행 상태를 저장하기 위한 시스템 호출의 커널 내부 함수(sys_backup())는 (그림 2)와 같이 설계한다.



(그림 2) sys_backup() 함수 구조

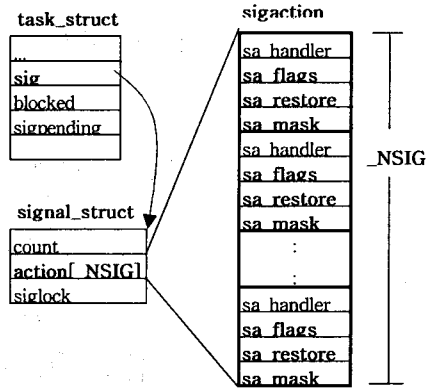
(그림 2)에서 sys_backup()은 단순히 검사점 플래그를 설정한다. 커널 모드에서 사용자 모드로 전환하기 위한 루틴(ret_from_sys_call)에 do_backup()을 추가함으로써 프로세스의 실행 상태를 저장하도록 한다. do_backup()에서는 현재 실행 중인 프로세스의 검사점 설정 플래그를 검사하여 설정되어 있을 경우, "[실행파일명].bak"이라는 복구 파일을 생성하여 프로세스의 실행 상태를 저장한다. 프로세스의 실행 상태를 복구하기 위하여 다음과 같은 정보들이 요구된다.

● 프로세스의 구조체(task_struct)

일반적으로 커널 내부에서는 프로세스의 모든 정보를 관리하기 위하여 프로세스 구조체(task_struct)를 유지한다. do_backup()에서 결함 허용 프로세스의 검사점 플래그 설정을 위한 필드를 프로세스 구조체에 추가한다.

● 프로세스 신호 정보(signal_struct)

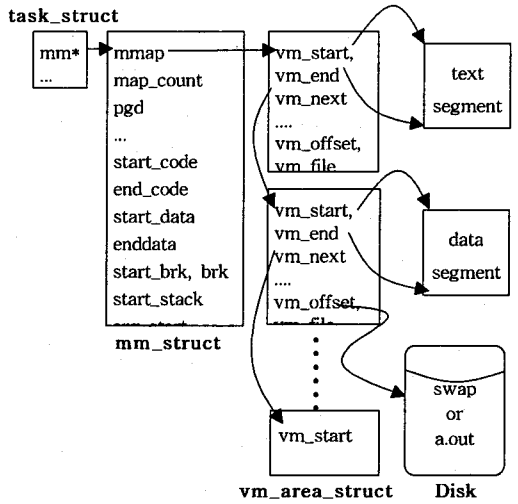
결함 허용 프로세스가 시스템 오류에 의해 종료될 경우 이에 대한 신호 정보를 저장한다. 프로세스의 신호 정보(signal_struct)는 (그림 3)과 같은 구조로 프로세스 구조체에 연결되어 있다.



(그림 3) signal_struct 연결 구조

● 프로세스의 주소 공간(mm_struct)

결함 허용 프로세스 복구를 위해 필요한 프로세스의 주소 공간(mm_struct)은 (그림 3)과 같은 구조로 프로세스 구조체에 연결되어 있다.

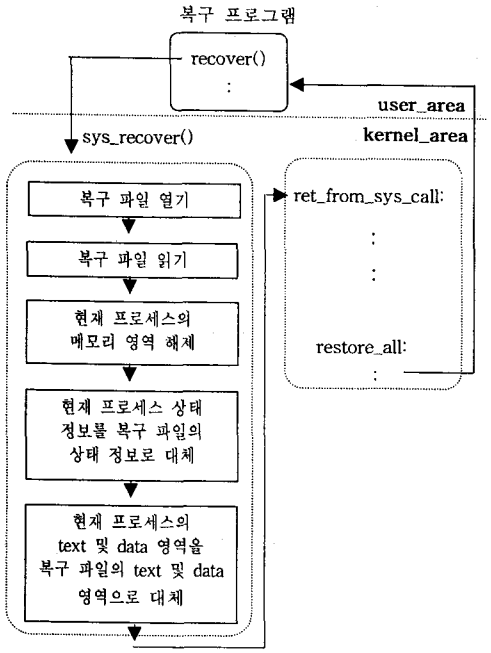


(그림 4) mm_struct 연결구조

결함 허용 프로세스의 복구를 위하여 프로세스의 text 및 data segment의 내용을 저장할 필요가 있다. (그림 4)에서 프로세스의 주소 공간과 관련된 정보는 프로세스 구조체 내부의 mm_struct 자료구조와 mm_struct 구조체의 mmap 포인터가 가리키는 vm_area_struct 리스트에 있다. vm_area_struct는 프로세스의 가상 메모리 영역으로써 여러 개의 segment로 구성되며, 결함 허용 프로세스의 복구를 위한 text 및 data segment가 저장된 메모리 주소는 vm_area_struct 구조체의 vm_start와 vm_end가 각각 가리킨다.

2.2 sys_recover()

결함 허용 프로세스의 실행 상태를 복구하기 위한 시스템 호출의 커널 내부 함수(sys_recover())는 (그림 5)와 같이 설계한다.



(그림 5) sys_recover() 구조

(그림 5)에서 sys_recover()는 복구 파일로부터 읽어들인 결함 허용 프로세스의 상태 정보와 text 및 data 영역의 내용을 읽어들인 후, 현재 실행 중인 복구 프로세스의 상태 정보와 text 및 data 영역의 내용을 복구 파일로부터 읽어들인 결함 허용 프로세스의 상태 정보와 text 및 data 영역으로 대체한다.

3. 구현 및 실험

본 논문에서는 두 개의 시스템 호출을 이용한 결함 허용 프로세스의 복구 기법을 제시하였다. 제시한 기법의 타당성을 검토하기 위하여 리눅스 커널 2.4.18 내부에 두 개의 시스템 호출을 구현하고, 이를 이용한 결함 허용 프로세스의 복구 기법을 실험하였다.

3.1 sys_backup() 구현

결함 허용 프로세스의 실행 상태를 저장하기 위한 시스템 호출(backup())을 지원하기 위하여 다음과 같이 커널 내부 함수들을 수정 및 추가하였다.

① 검사점 플래그 추가 및 설정

결함 허용 프로세스를 식별하기 위한 검사점 플래그(int need_bak)를 task_struct에 추가하고, 현재 프로세스에 대한 검사점 플래그 값을 TRUE로 설정하는 sys_backup()를 정의한다.

```
struct task_struct {
        :
      int need_bak; /* 검사점 플래그 추가*/
};
sys_backup() {
      /*검사점 플래그 추가*/
        current->need_bak=TRUE;
}
```

② ret_from_sys_call 수정

시스템 호출로부터 사용자 모드로 복귀하는 루틴(ret_from_sys_call)에 프로세스 실행 상태 정보 저장을 위한 do_backup()을 다음과 같이 추가한다.

```
ENTRY(ret_from_sys_call)
        :
       jne signal_return
       call SYMBOL_NAME(do_backup)  /* do_backup() 추가*/
restore_all:
       RESTORE_ALL
```

③ do_backup() 추가

프로세스 실행 상태 정보 저장을 위하여 다음과 같은 기능을 수행하는 do_backup() 추가한다.

```
do_backup() {
    make_file();  /*복구 파일명 부여 및 생성*/
    write_file();  /*프로세스의 실행 상태 저장*/
}

make_file() {
   char *extend = ".bak";
        memcpy(filename, current->comm, k);
        filename[i++]=extend[i++];
        filp_open(filename, O_CREAT, ...);
            :
}

write_file() {
        write(file,(char*)current, ...);
        write(file,(char*)current->sig, ...);
        write(file,(char*)&vm_area_count, ...);
        write(file,(char*)current->mm, ...);
        write(file,(char*)regs, ...);
        write(file,(char*)current->mm->mmap, ...);
        write(file,(char*)current->mm->mmap->vm_start,);
            :
}
```

3.2 sys_recover() 구현

결함 허용 프로세스의 실행 상태를 복구하기 위한 시스템 호출(recover())을 지원하기 위하여 다음과 같이 정의된 커널 함수들을 추가하였다.

```
sys_recover() {
        :
   filename=getname(name); /*파일명 가져오기*/
   filp_open(filename, O_RDONLY,0600); /*파일열기*/
/* 복구 상태 정보를 읽기 */
   read(file,(char*)&task,n,&file->f_pos);
   read(file,(char*)&sig,n,&file->f_pos);
   read(file,(char*)&vm_area_count,n,&file->f_pos);
   read(file,(char*)&mm,n,&file->f_pos);
   read(file,(char*)&regs,n,&file->f_pos);
/* 현재 프로세스의 메모리 영역 해제 */
   flush_old_exec(&bprm);
/* 결함 허용 프로세스의 상태 정보 복구 */
   restore_task(&task);
   restore_mm(&mm);
   restore_sig(&sig);
/* 결함 허용 프로세스의 text 및 data 복구 */
   read(file,(char*)&tmp,n,&file->f_pos);
   do_mmap(NULL,tmp,vm_start, ...);
   find_vma(current->mm,tmp.vm_start);
   read(file,(char*)tmp.vm_start,n,&file->pos);
   putname(filename);
}
```

결함 허용 프로세스의 복구는 이전 프로세스의 상태정보를 복구 프로세스의 상태정보에 매핑 시킴으로써, 이전의 프로세스 상태를 복구하여 복구파일에 저장된 실행 상태의 시점에서부터 프로그램을 계속 수행할 수 있도록 한다.

3.3 응용 프로그램 및 복구 실험

본 논문에서 제시한 결함 허용 프로세스의 타당성을 검토하기 위하여 backup() 및 recover()를 리눅스 커널 2.4.18 내부에 추가하고 이를 이용한 다음과 같은 응용 및 복구 프로그램을 작성하여 실험하였다.

● 응용 프로그램(test.c)

```
/* backup 시스템호출의 정의 */
_syscall1(int, backup, int, pid);
/* 계산 수행 */
for (i=0; i<100; c++)
{
        for (j=0; j<100; j++)
           {
           array[ i ][ j ] = i + j;
           printf("%d\n", array[ i ][ j ]);
           }
}
/* backup() 시스템 콜 함수 호출 */
backup(-1);
}
```

● 복구 프로그램(recover.c)

```
/* recover 시스템호출의 정의 */
_syscall1(int, recover, char*, name);
int main(int argc, char** argv) {
                :
/* recover() 시스템 콜 함수 호출 */
          recover(argv[1]);
          return -1;
}
```

① 검사점 파일 생성

응용 프로그램(test)이 실행되는 중간에 "ctrl+C "로 비정상 종료시킨 후, 검사점 파일(test.bak)이 생성됨을 확인한다.

```
[root@localhost backup]# ls
test   test.c   test.bak   recover.c   recover
```

② 결함허용 프로세스 복구

복구프로그램인 "recover"로 비 정상적으로 종료되었던 결함 허용 프로세스를 복구 시킨다.

```
[root@localhost backup]# ./recover test.bak
```

4. 결론

프로세스의 복구 기법은 장시간 실행을 요하는 프로세스에서 하드웨어, 소프트웨어적 결함으로 발생하는 심각한 피해를 최소화하기 위하여 절대적으로 요구된다. 프로세스 복구를 위한 대부분의 기존 기법들은 결함 발생의 원인 규명과 예측에 의한 오버헤드가 있다.

본 논문에서는 이러한 오버헤드를 최소화하기 위한 결함 허용 프로세스 복구 기법을 제시하였다. 제시한 기법은 사용자의 편의성과 시스템의 효율성을 제공하기 위하여 프로세스 상태저장 및 복구 기능을 커널 내부에 구현하였다.

참고문헌

[1] 조유근, 최종무, 홍지만 저 "리눅스 매니아를 위한 커널 프로그래밍" 교학사.

[2] 임성락 저 "운영체제"

[3] 다니엘 보베이, 마프코 체사티 저 "리눅스 커널의 이해" 한빛 미디어

[4] Keith Haviland, Dina Gray, Ben Salama 저 "UNIX system programing" 홍릉 과학 출판사