Chungnam National University
Department of Computer Science and Engineering

2013년 가을학기

기말고사

2013년 12월 3일
시스템 프로그래밍

분반/학번	반 번
이름	

문제	배점	점수
1	16	
2	16	
3	20	
4	10	
5	20	
6	20	
총계	102	

나는 이 답안을 부정행위 없이, 내 스스로의 힘으로 작성하였으며, 다른 학생의 것을 보거나, 다른 학생에게 보여주지 않았음을 선서합니다. ( )

문제 1. 기초지식 (16점)

1) (4점) 좀비 프로세스란 무엇이며, 좀비가 많으면 무슨 문제가 생기는가?

- 좀비 : 종료되었지만 제거되지 않은 프로세스
- 메모리를 낭비한다.

2) (4점) 시그널은 1만 사이클 이상이 소요되는 매우 오버헤드가 큰 작업이다. 그 이유로는 시스템콜의 처리, 운영체제에 의한 프로세스의 실행 정지, 문맥전환(context switch)등을 들 수 있는데, 특히 컨텍스트 교체작업시에 메모리 접근 성능이 대폭 나빠질 가능성이 있다. 그 이유는 무엇인지 설명하시오. (힌트. 캐시 메모리와 관련됨)

- 캐시를 오염시켜서 캐시미스를 발생시킨다. 캐시미스가 발생하면 외부의 메인메모리를 접근해야 하므로 실행속도가 급격히 느려진다

3) (4점)longjump( )와 같은 비지역성 점프 함수는 어떤 경우에 사용하는 것이 유리한가?

- 에러처리와 같이 규모가 큰 코드에서 직접 점프나 프로시져 콜로는 접근할 수 없는 장거리 코드영역의 점프에 사용

4) (4점) 내부단편화(Internal fragmentation)를 줄이기 위해 malloc( )을 구현할 때 사용할 수 있는 방법을 한 가지만 설명하시오.

- 블록 쪼개기

문제 2. (16점) [시그널] 아래 프로그램은 자식프로세스를 생성하여 date 란 프로그램을 지속적으로 실행시키는 프로그램이다. 부모 프로세스는 자신의 자식 프로세스들을 joblist 로 관리하며, 하나의 자식 프로세스에 대해 한 개의 job 엔트리를 추가한다. addjob()과 deletejob()은 이 joblist로부터 엔트리를 추가하거나 삭제한다. 부모프로세스는 실행된 프로그램의 job 들을 관리하고 자식이 종료되었을 경우 자식을 청소하는 코드로 구성되어 있다.

```
void handler(int sig){
    pid_t pid;
    while ((pid = waitpid(-1, NULL, 0)) > 0)
        deletejob(pid);
}
int main(int argc, char **argv)
{
    int pid;

    signal(SIGCHLD, handler);
    initjobs();

    while (1) {
        if ((pid = Fork()) == 0) {
            Execve("/bin/date", argv, NULL);
        }
        addjob(pid);
    }
    exit(0);
}
```

1) (4점) 위 SIGCHLD 핸들러에서 waitpid( ) 함수가 하는 역할이 무엇인지 설명하시오.

- 좀비를 제거한다.

2) (4점) 위 SIGCHLD 핸들러에서 발생한 SIGCHLD를 모두 처리해야 하기 위해 while 루프를 이용해야 하는 이유를 설명하시오.

- 연속적으로 발생하는 시그널을 놓치지 않고 모두 처리하기 위해. 시그널은 큐에 저장되지 않으므로… 등등

3) (4점) 여러분이 이미 실습시간에 경험했듯이 위 프로그램을 실행시키면 race condition 이라는 미묘한 오류가 발생할 수 있다. 위 프로그램을 실행시켰을 때, race condition에 의해 어떤 오류가 발생하는지 설명하시오.

- 부모프로세스가 addjob 을 실행하기도 전에 자식프로세스가 종료되어 SIGCHLD가 발생하여 없는 job을 deletejob 하려고 한다.

4) (4점) 위 프로그램이 정상적으로 실행되도록 하기 위해 02반의 이XXX 학생이 다음과 같이 코드를 수정하였다. 이렇게 프로그램을 수정하면 racing condition이 없어지게 되는 이유를 설명하시오.

```
int main(int argc, char **argv)
{
    int pid;
    sigset_t mask;

    signal(SIGCHLD, handler);
    initjobs();

    while (1) {
          Sigemptyset(&mask);
          Sigaddset(&mask, SIGCHLD);
          Sigprocmask(SIG_BLOCK, &mask, NULL);
          if ((pid = Fork()) == 0) {
              Sigprocmask(SIG_UNBLOCK, &mask, NULL);
              Execve("/bin/date", argv, NULL);
          }
          addjob(pid);
          Sigprocmask(SIG_UNBLOCK, &mask, NULL);
    }
    exit(0);
}
```

- 부모 프로세스가 addjob을 한 이후에야 SIGCHLD 시그널을 unblock시켜 deletejob은 반드시 addjob 이후에 실행되게 된다.

문제 3. (32점) **[동적메모리 할당]** 다음과 같은 정렬 요건과 블록구조 조합을 이용하는 메모리 할당기를 설계하려고 한다. 직접가용리스트(Explicit list)를 사용하며, 가용블록 내에는 4바이트의 pred와 succ 포인터를 포함하며, 데이터(payload)의 크기는 0바이트는 허용하지 않는다. 헤더와 풋터는 4바이트씩 사용한다.

구현방법	정렬규칙	할당블록	가용블록
1	Double word	Header and footer	Header and footer
2	Double word	Header, no footer	Header and footer

1) (4점) 1번 구조를 이용하는 경우, 할당된 블록의 최소 크기와 가용블록의 최소 크기는 각각 몇 바이트인가? 계산 과정을 설명해야 한다.

- 할당된 블록 : 헤더(4)+풋터(4)+데이터(1)+패딩(7) = 16바이트
- 가용블록 : 헤더(4)+풋터(4)+pred(4)+succ(4) = 16바이트

2) (4점) 1)번 문제의 결과로부터 1번 구조를 이용하는 메모리 할당 라이브러리를 구현하려면, 최소 블록의 크기는 몇 바이트가 되어야 하는가?

- 최소 블록의 크기는 할당된 블록과 가용블록의 최소 크기중 최대값으로 결정한다. 이 경우는 16바이트

3) (4점) 2번 구조를 이용하는 경우, 할당된 블록의 최소 크기와 가용블록의 최소 크기는 각각 몇 바이트인가? 계산 과정을 설명해야 한다.

- 할당된 블록 : 헤더(4)+데이터(1)+패딩(3) = 8 바이트
- 가용블록 : 헤더(4)+풋터(4)+pred(4)+succ(4) = 16 바이트

4) (4점) 2번 구조를 이용하는 메모리 할당 라이브러리를 구현하려면, 최소 블록의 크기는 몇 바이트가 되어야 하는가?

- 16 바이트

5) (4점) 위에서 계산해 본 것처럼, 어떤 동적메모리 할당기의 구현에서 최소블록의 크기가 크게 되면, 작은 경우보다 어떤 단점이 생기는가?

– 내부 단편화가 커진다. 메모리 활용도가 나빠진다.

6) (4점) 양방향 포인터를 사용하는 직접리스트(Explicit list) 방식으로 malloc( )을 구현할 때 free블럭을 검색하는 속도를 간접리스트(Implicit list) 방식에서와 비교할 때 어느 쪽이 더 빠른지 설명하고, 그 이유를 설명하시오.

- 직접리스트가 더 빠르다. 그 이유는 직접리스트는 가용블록들만 검색하기 때문이다.

7) (4점) Segregated list(분리 리스트) 방식을 사용하는 경우 할당되었던 메모리 블록을 반환할 때, 해당 크기 클래스의 가용리스트의 제일 앞에 반환하는 방법을 LIFO 방식이라고 부른다. LIFO방식의 반환 알고리즘 구현시 장점을 설명하시오.

- 반환시 시간 복잡도가 상수다.. 따라서 처리 시간이 빠르다.

8) (4점) Segretated list 방식을 사용하는 경우 malloc 할당할 때 소요되는 시간 복잡도는 얼마인가?

– 크기별 할당하게 되므로, 할당 소요시간이 상수다

문제 4. [가비지컬렉터] (10점)

1) (4점) 자바나 ML언어등에서 사용하고 있는 가비지 컬렉터란 무엇인지 설명하시오.

- 프로그램에서 더 이상 사용하지 않는 할당된 메모리를 반납해주는 동적 할당기

2) (6점) 가비지 컬렉터를 구현하는 알고리즘으로 Mark & Sweep 방식에 대해 설명하시오.

- 프로그램이 실행하는 동안 접근하는 메모리들의 상관 관계를 접근그래프로 표시하고, mark 알고리즘을 적용해서 접근가능 노드들을 표시하고, sweep 알고리즘을 이용해서 표시가 되지 않은 할당된 노드들은 반납시켜준다.

문제 5. (16점) [4지선다]

1) malloc 함수가 내부에서 사용하는 메모리블록 헤더의 크기를 무작정 줄이면, 어떤 일이 발생하는가? (1)

(1) 내부 단편화가 줄어든다 (2) 내부단편화가 늘어난다 (3) 외부단편화가 줄어든다 (4) 외부단편화가 늘어난다.

2) 에러는 발생하지 않는다고 가정할 때, 다음 함수들 중에서 리턴이 정확히 한번만 일어나는 함수는 무엇인가? (5)

(1) fork( ) (2) execve( ) (3) exit( ) (4) longjump( ) (5) waitpid( )

3) 어떤 프로그램이 SIGCHLD와 SIGUSR1 을 블록하고 있다. 그리고 나서 SIGCHLD 한 개, SIGUSR1 한 개, 그리고 또 다른 SIGCHLD 한 개가 다시 도착한다고 하자. 이 프로그램이 이 모든 시그널들을 unblock 시킨 후에 어떤 시그널들을 이 프로그램이 처리하게 되겠는가? (3)

(1) 아무것도 처리하지 않는다. 왜냐하면 블록된 시그널들은 모두 없어지기 때문이다

(2) 한 개의 SIGCHLD만 처리한다. 왜냐하면, 이후에 도착하는 모든 시그널은 버리기 때문이다.

(3) 한 개의 SIGCHLD와 한 개의 SIGUSR1 만 처리한다. 왜냐하면, 두번째 SIGCHLD는 버리기 때문이다.

(4) 세 개 모두 처리한다. 왜냐하면 시그널은 절대로 버리지 않기 때문이다.

4) 동적메모리 할당을 이용하는 이유는 무엇인가? (3)

(1) 힙이 스택보다 훨씬 더 빠르기 때문에

(2) 스택이 버퍼 오버플로우로 인한 에러 가능성이 있기 때문에

(3) 스택에 데이터를 저장하려면 컴파일 시에 데이터의 크기를 알아야 하기 때문에

(4) 답이 없다

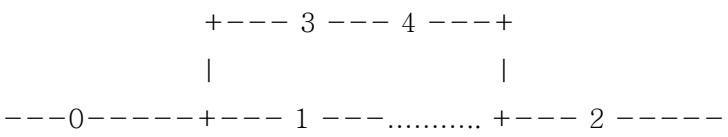
문제 6. [동시성 프로세스] (20점) 다음과 같은 프로그램을 보고 질문에 답하시오.

```
int main()
{
    int val = 2;

    printf("%d", 0);
    fflush(stdout);

    if (fork() == 0) {
        val++;
        printf("%d", val);
        fflush(stdout);
    }
    else {
        val--;
        printf("%d", val);
        fflush(stdout);
        wait(NULL);
    }
    val++;
    printf("%d", val);
    fflush(stdout);
    exit(0);
}
```

1) (5점) 위 프로그램을 실행시킬 때 가능한 출력들을 알아보기 위해, 프로세스 그래프를 그리시오.



2) (5점) 위 프로그램으로 출력 가능한 문자열의 종류는 모두 몇 가지인가?

- {0, {(3, 4), 1}, 2 } 형태로 출력되어야 한다. 따라서, (3, 4)와 1이 실행순서가 바뀔수 있으므로, 03412, 01342 의 2가지만 가능하다

3)(10점) 아래의 (A)~(E)의 5개의 출력이 위 프로그램으로부터 출력될 수 있는지, 없는지 판단하시오.(각2점, 오답은 -1점)

(B), (E) 만 가능하고, 나머지는 불가능

(A) 01432 (1) 있다 (2) 없다
(B) 01342 (1) 있다 (2) 없다
(C) 03142 (1) 있다 (2) 없다
(D) 01234 (1) 있다 (2) 없다
(E) 03412 (1) 있다 (2) 없다