**CSED211 Homework6**

20210774 김주은

**1. Exercise 9.15 on page 915 with the following requests.**

먼저, block size는 8바이트의 배수여야 하므로 제일 인접한 8byte의 배수로 반올림한 값이 block size가 된다. 그 다음 block header의 경우 상위 비트는 block size이고, 하위 비트는 블록이 할당되었음을 나타낸다.

malloc(3)의 경우 0x8 | 0x1 이므로 0x9이다.

malloc(15)의 경우 15+4 이상의 8의 배수는 24이므로, 0x18 | 0x1 이므로 0x19이다.

malloc(25)의 경우 0x20 | 0x1 이므로 0x21이다.

malloc(40)의 경우 0x28 | 0x1 이므로 0x29이다.

Answer:

|  |  |  |
| --- | --- | --- |
| Request | Block size(decimal bytes) | Block header(hex) |
| malloc(3) | 8 | 0x9 |
| malloc(15) | 24 | 0x19 |
| malloc(25) | 32 | 0x21 |
| malloc(33) | 40 | 0x29 |

**2. Exercise 9.16 on page 915**

Explicit free list를 사용하며, 4-byte의 pred and succ pointer을 사용한다.

즉, 4byte 각각의 pred and succ pointer을 free block내에 가지고, 데이터의 크기도 0보다 커야 한다.

Single word 정렬을 따르고, allocated 와 free block이 모두 header 와 footer을 가지는 경우에서는allocated block은 4byte짜리의 header와 footer을 포함하는 동시에 데이터의 크기도 0보다 커야하므로 최소 block size가 8byte가 이 될수는 없다. Free block의 경우에는 총 8byte를 차지하는 header와 footer와 함께, 4byte짜리의 pred and succ pointer가 포함되기 때문에 최소 16byte가 필요하다.

Single word 정렬을 따르고 allocated block이 footer을 가지지 않는 경우에도 free block은 위의 경우와 똑같이 4byte짜리의 header, footer, pred pointer, succ pointer을 포함하므로 최소 16byte가 필요하다.

Double word 정렬은 같은 조건에서의 위의 Single word 정렬에서 모두 최소 block size가 16byte이므로 따로 정렬을 한 번 더 해줄 필요없이 모두 16으로 동일하다.

Answer:

|  |  |  |  |
| --- | --- | --- | --- |
| Alignment | Allocated block | Free block | Minimum block size(bytes) |
| Single word | Header and footer | Header and footer | 16 |
| Single word | Header, but no footer | Header and footer | 16 |
| Double word | Header and footer | Header and footer | 16 |
| Double word | Header, but no footer | Header and footer | 16 |

**3. Exercise 9.19 on page 916**

3-1.

(a) In a buddy system, up to 50% of the space can be wasted due to internal fragmentation.

(b) The first-fit memory allocation algorithm is slower than the best-fit algorithm (on average).

(c) Deallocation using boundary tags is fast only when the list of free blocks is ordered according to increasing memory addresses.

(d) The buddy system suffers from internal fragmentation, but not from external fragmentation.

1번의 경우에는 (a)가 true statement이다.

(b) first-fit의 경우 free list를 처음부터 검색해서 가용블록을 선택하는 알고리즘이라면 best-fit은 모든 가용 블록을 검사하는 알고리즘이기 때문에 best-fit algorithm이 평균적으로 더 느리므로 false statement이다.

(c) free block의 list가 주소 순으로 관리되지 않더라도 LIFO 순으로 유지한다면, 경게태그를 사용하는 deallocation을 빠르게 할 수 있다. 그러므로 c도 false statement이다.

(d) external fragmentation도 발생하기 때문에, false statement이다.

3-2.

(a) Using the first-fit algorithm on a free list that is ordered according to decreasing block sizes results in low performance for allocations, but avoids external fragmentation.

(b) For the best-fit method, the list of free blocks should be ordered according to increasing memory addresses.

(c) The best-fit method chooses the largest free block into which the requested segment fits.

(d) Using the first-fit algorithm on a free list that is ordered according to increasing block sizes is equivalent to using the best-fit algorithm.

2번의 경우에는 (d)가 true statement이다.

(a) first-fit은 리스트 처음부터 탐색하기 때문에 decreasing block size로 정렬될 경우 allocation을 할 때 훨씬 효율적이다.

(b) 주소 순으로 블록 리스트를 정렬하는 것은 best-fit method가 아니라 first-fit method에 대해서 검색 시간을 줄일 수 있다.

(c) best-fit method는 largest free block을 고르는 것이 아니라, smallest free block을 고른다.

3-3. Mark & Sweep garbage collectors are called conservative if

(a) They coalesce freed memory only when a memory request cannot be satisfied.

(b) They treat everything that looks like a pointer as a pointer.

(c) They perform garbage collection only when they run out of memory.

(d) They do not free memory blocks forming a cyclic list.

(b) 가 true statement이다. Pointer가 아닌 것들도 pointer로 위장하게 되면, unmarked되어야 하는 block이 marked되는 보수적인 경우가 생기게 되는 것이다.

**4. Exercise 10.6 on page 950.**

Open함수는 항상 프로세스 내에서 현재 열려 있지 않은 가장 작은 식별자를 리턴하므로 open으로의 첫번재 호출은 식별자 3을 return하고, 두번 째 호출은 식별자 4를 return한다. 이후에 식별자 4에 해당되는 파일이 다시 close되었다가 open되므로 마지막 open함수는 식별자 4를 return할 것이다. 그러므로, 출력은 fd2 = 4가 된다.

Answer : fd2 = 4

**5. Exercise 10.9 on page 950**

기존에 Fstatecheck가 실행되면 자동적으로 dup2함수가 불리면서 descriptor table entry간의 복제가 이루어진다. < foo.txt의 명령에 의해 표준 입력에서 foo.txt로 입력이 바뀌어야 하므로, 표준 입력에 해당하는 식별자 0이 file table에 있는 foo.txt 파일에 해당하는 entry를 가리키게 되는 것이다. 그러고 fstatecheck가 동작하면 식별자 3에 대한 메타데이터를 분석하여 출력하는데, 여기서 식별자 3이 가리키는 파일 entry는 foo.txt와 대응되지 않으므로, bad file descriptor 오류를 내는 것이다.

그러므로, execve 함수 호출전 식별자 3에 foo.txt를 열어서 가리키게 하고, 식별자 0도 foo.txt를 가리키게 함으로써 execve호출 이후 fstatcheck에서 식별자 3에 대한 메타데이터를 잘 출력할 수 있도록 한다. 그래서 open함수를 통해 먼저, foo.txt를 열고 fd 3이 가리키도록 하며 이를 STDIN\_FILENO라는 fd 0에 복제함으로써 fd 0이 foo.txt를 가리키도록 해준다. 그리고 close(fd)를 통해 fd 3이 foo.txt를 더이상 가리키지 않도록 한다.

Answer :

If(Fork() == 0)

{

fd = Open(“foo.txt”, O\_RDONLY, 0);

Dup2(fd, STDIN\_FILENO);

Close(fd);

Execve(“fstatcheck”, argv, envp);

}

**6. Enumerate the difference between process and thread concisely.**

1) Process

프로세스의 겨우 각 논리적 흐름을 커널이 스케쥴하고 관리하는 메커니즘이라고 할 수 있다. 그러므로, 프로세스의 경우 별도의 독립적인 가상 주소공간과 programmer visible state를 가진다. 그러나 별도의 주소공간을 사용하면서 프로세스가 상태 정보를 공유하는 것이 어려우므로, 명시적인 프로세스 간 통신 메커니즘이라고 불리는 IPC 메커니즘을 사용해야 한다.

2) Thread

쓰레드는 한 개의 process context에서 돌아가는 논리흐름으로 커널에 의해 스케쥴되는 형식이다. 각 쓰레드는 고유의 정수 쓰레드, 스택, 스택 포인터, 프로그램 카운터, 범용 레지스터, 조건 코드를 포함하는 자신만의 쓰레드 컨텍스트를 가진다. 그리고 한 개의 프로세스에서 돌고 있는 모든 쓰레드는 이 프로세스의 전체 가상 주소를 공유한다. 즉, 프로세스처럼 커널에 의해 자동으로 스케줄되지만 다수의 쓰레드는 한 개의 process context에서 돌아가므로 이 프로세스의 가상 주소 공간의 전체 내용을 공유하는 것이다.

**7. Exercise 12.6 on page 1031**

A.

|  |  |  |  |
| --- | --- | --- | --- |
| Variable  instance | Referenced by | | |
| Main thread? | Peer thread 0? | Peer thread 1? |
| ptr | Yes | Yes | Yes |
| cnt | No | Yes | Yes |
| i.m | Yes | No | no |
| msgs.m | Yes | Yes | Yes |
| myid.po | No | Yes | no |
| myid.p1 | No | No | yes |

스택 변수들은 사적 변수이고, 전역 및 정적 변수들은 공유 변수다. cnt같은 정적 변수들은 공유가 자신들의 영역 내의 함수들로 제한되기 때문이다.

ptr : 전역변수이다. ptr=msgs; 구문으로 인해 메인 쓰레드에 의해 쓰여지고, 피어 쓰레드에서 ptr[myid]로 읽어준다. 그러므로 모두 YES이다.

cnt: 함수 안에서 static으로 선언되었다. 단 한 개의 인스턴스를 갖는 정적 변수이며, 두 개의 피어 쓰레드가 읽고 쓴다. 그러므로 No, yes, yes이다.

i.m: 메인 쓰레드의 스택에 저장된 지역 자동 변수이다. 피어 쓰레드로 값이 전달되지만 피어 쓰레드에서는 스택에서 참조하지 않기 때문에 피어 쓰레드에서 공유되지 않는다. 그러므로, yes, no, no 이다.

msgs.m: 메인 쓰레드의 스택에 저장되는 지역 자동 변수이다. 또한, i.m과는 다르게 두 개의 피어 쓰레드에 의해 ptr을 통해 간접적으로 참조되므로, yes,yes,yes이다.

myid.po 과 myid.p1: 지역 변수이므로, 두 개의 인스턴스를 갖는다. 하나는 피어 쓰레드 0의 스택에, 다른 하나는 피어 쓰레드 1의 스택에 있다. 그러므로, myid.po는 no, yes, no이고, myid..p1은 no,no,yes이다.

B. 변수 ptr, cnt, msgs는 한 개 이상의 쓰레드가 참조하기 때문에 공유되어 있다.

**8. Exercise 12.17 on page 1067**

A.

문제는 main thread가 peer thread가 끝나기까지 기다리지 않고 종료하기 때문이다. Exit를 호출하게 되면 전체 프로그램이 종료하게 되는데, 이 때 전체 프로그램을 종료하기 때문에 실행 중인 다른 모든 쓰레드도 강제 종료되는 것이다. 그러므로 peer thread는 “Hello, world!”를 출력하기 전에 종료되어 아무것도 출력되지 않는 것이다.

B.

코드의 10번 줄에 있는 exit(0); 코드를 다른 pthread함수를 호출하여 해결할 수 있다. pthread\_exit이나 pthread\_join 함수를 호출하면 된다. pthread\_exit함수의 경우 다른 모든 쓰레드가 종료하기를 기다리며, pthread\_join 함수도 마찬가지로 다른 쓰레드가 종료되기를 기다리기 때문에 사용하면 버그를 해결할 수 있다. Ex) pthread\_exit(NULL);