03/31 운영체제 과제

Bus와 Lan의 속도 비교

- 가장 속도가 빠른 Bus는 PCIe 5.0 128(x16)으로, 32.0GHz의 전송속도를 가진다.
- 가장 속도가 빠른 Lan은 10 Gigabit Ethernet (10GBASE-X)으로, 1Gbps의 전송속도를 가진다.

Bus와 Lan의 대역폭 비교

- PCIe 5.0 128(x16) BUS의 대역폭은 128GB/s입니다.
- 10 Gigabit Ethernet (10GBASE-X)의 대역폭은 1.25GB/s입니다.

Bus와 Lan의 Naming방식 비교

- Bus는 다음 3가지로 구성된다.

  1. Address : 데이터 전송을 할때 데이터가 메모리 어디에 있는지를 나타내는 위치

  2. Data : 연산된 결과와 같이 의미있는 값을 말한다.

  3. Control : 앞의 주소 정보와 실제 데이터가 컴퓨터 시스템안에서 어떻게 관리되고 방향으로 전송되어야 하는지

  만약 하드 디스크에 저장되어 있는 MP3 파일의 음악을 듣을려고 한다고했을 때, 그 mp3 데이터가 스피커를 통해 소리를 내기 위해서는 사운드 카드로 전송되어야 한다. 여기서 MP3 파일이 위치한 하드 디스크와 사운드 카드가 주소 정보라고 생각하면 된다. 즉 computer의 각 컴포넌트들은 Address bus의 주소 정보로써 naming된다.

- LAN은 MAC주소를 이용하여 naming을 하는데, 이때 MAC주소란 데이터 링크 계층에서 통신을 위한 네트워크 인터페이스에 할당된 고유 식별자 이다. MAC주소는 일반적으로 제조업체가 거의 하드웨어 자체에 이식해버리기에 programmable하지 않다.

4.7 수업시간 과제

2018320161 송대선

Q. CPU, memory, disk의 속도를 각각 알아오기

1. CPU

제 PC의 CPU 속도는 1.99GHz으로,

이는 하나의 코어에 대하여 시간당 클럭이 약 552,777번 진동한다는 의미입니다.

제 CPU의 코어는 4개이므로 시간당 클럭이 약 2,211,108번 진동한다는 의미입니다.

2. Memory

제 PC의 메모리 속도는 2133MHz으로,

이는 시간당 약 592,500번 진동한다는 의미입니다.

3. Disk

제 PC의 SSD 읽기 속도는 약 3420.45(MB/s), 쓰기속도는 약 1375.75MB/s으로,
이는 읽기속도 약 0.950125(MB/h), 쓰기속도 약 0.3821527777777778(MB/h)
즉, 읽기속도 약 972.928(KB/h), 쓰기속도 약 391.324444444444(KB/h)임을 의미합니다.

4_9 수업시간 과제

2018320161 송대선

a.out

a.out이라는 이름은 어셈블러 출력(assembler output)을 줄인 말이다.

gcc 컴파일러로 c언어 파일을 컴파일하면, a.out이라는 실행 파일이 생성되고, 생성된 a.out 파일을 이용하여 실행 할 수 있는 구조이다. 즉, a.out이란 실행가능한 프로그램이라는 것이다.

a.out의 구조는 7개의 section으로 나뉘는데 각각, exec header section, text segment section, data segment section, text relocations section, data relocations section, symbol table section, string table section이다.

COFF

COFF라는 이름은 Common Object File Format의 약자이다.

대부분의 Microsoft EXE과 DLL 파일들은 COOFF기반의 format을 사용한다.

Microsoft's Portable Executable (PE/COFF) standard가 COFF과 가장 널리 호환되는 포맷이다.

COFF 파일의 구조는 File Header, Optional Header, Section Header, Section Relocation Table, Section Line Number Table, Symbol Table, String Table으로 구성된다.

4_28수업시간과제

1. longjmp()와 setjmp() 함수가 무엇인지 설명하고, signal과의 관계를 설명하시오.

setjmp()와 longjmp()는 C언어의 goto문과 유사하지만 다소 차이가 있습니다.

int setjmp(jmp_buf env)

- 함수가 호출되는 순간 스택값들은 env에 저장됩니다.

- setjmp()호출, longjmp할 곳을 지정합니다.

void longjmp(jmp_buf env, int val)

- longjmp를 호출하면 setjmp()를 한 곳으로 돌아갑니다.

setjmp와 longjmp는 goto와 비슷합니다.
setjmp로 돌아와야 할 부분을 설정하고 longjmp로는 goto문을 나타냅니다.
c에서는 함수간의 goto는 허용되지 않습니다. 하나의 함수 안에서만 허용이 됩니다.
그래서 setjmp를 호출하여 내가 다음에 올 곳 이라고 명시를 하고 longjmp를 호출함으로써 다시 setjmp를 호출한 곳으로 오게 됩니다.
goto와 다르게 한 함수안에서 적용되는것이 아닙니다.

longjmp() 함수는 setjmp() 함수에서 이전에 env에 저장한 스택 환경을 복원합니다.

setjmp()과 longjmp() 함수는 비로컬 이동을 수행하기 위한 방법을 제공합니다.

종종 signal핸들러에서 사용됩니다.

2. 07 IPC 강의자료에서 18p의 시그널 예제를 실행해보시기 바랍니다.

다음과 같은 결과를 얻었습니다.

```
song@song-VirtualBox:~/assignment/ass_04_28$ ./signal
signal handler
song@song-VirtualBox:~/assignment/ass_04_28$
```

```
song@song-VirtualBox:~/assignment/ass_04_28$ sudo vim signal.c
```

```
song@song-VirtualBox:~/assignment/ass_04_28$ sudo gcc signal.c -o signal
signal.c: In function ‘main’:
signal.c:27:1: warning: implicit declaration of function ‘sigpause’; did you mea
n ‘pause’? [-Wimplicit-function-declaration]
 sigpause(SIGUSR1);
 ^~~~~~~~
 pause
song@song-VirtualBox:~/assignment/ass_04_28$
```

5_7 과제

Q. 한 프로세스에 Multi Thread(4개의 스레드) 를 사용한다고 했을때, 각 스레드에 Time Quantum은 어떻게 계산될지 조사 및 생각 해보세요

Multi Thread에 대한 Time Quantum, 즉 thread quantum을 system에서 정의하는 방법으로는 the amount of time that the schedule allows a thread to run before scheduling a different thread

Thread Quantum이란 같은 우선순위의 Process들의 구동될 경우 하나의 Process에 최소한의 CPU점유 시간을 의미한다.

시스템상에 복수의 실행 가능한 thread가 있는 경우 스케쥴러는 정해진 짧은 시간 간격으로 나누어서 thread를 CPU에 순서대로 할당하거나 바꾸면서 실행시킨다. 이 쓰레드를 실행시키는 짧은 시간을 "퀀텀 타임(quantum time)"이라고 부른다.

동일한 process에 속한 user thread들끼리 CPU를 경쟁하기 때문에 Process-Contention Scope(PCS)가 발생한다. 따라서 thread library를 이용해 scheduling 및 관리돼야 한다.

보다 구체적인 scheduling읏 여러 thread를 하나로 묶어 사용하거나 혹은 각각의 thread에게 일을 분배하는 방법을 생각해볼 수 있을 것 같다. 실제로 사용되는 모델로는 Many to One Model, One to One Model, Many to Many Model 등이 있습니다.

Many to One Model은 N개의 User-thread가 1개의 Kernel thread 공유하는 모델로, User thread는 Kernel thread를 공유하므로 효율적이나 Kernel이 하나이므로 multi 효과가 없습니다. One to One Model은 1:1로 대응하여 User thread와 Kernel thread가 N개씩 같은 수를 가집니다. 병행성 효과는 좋으나 User가 Kernel thread를 과다하게 생성하는 문제가 발생하여 과하게 Resource룰 사용합니다. Many to Many Model은 앞의 Many to One Model과 One to One Model을 합친 것으로 User thread가 Kernel thread를 무작정 만드는 것이 아니라 Kernel에 갯수제한을 두어 만드는 모델입니다.

05_19_과제

2018320161_컴퓨터학과_송대선

int is_wrt //버퍼에 접근하는 Writer 숫자

semaphore rea//Reader 간 동기화

초기값 : is_wrt=0

Writer 프로세스

```
is_wrt = 1
wait(wrt); // entry section
…
writing is performed
…
signal(wrt); //exit section
is_wrt = 0
```

Reader 프로세스

```
wait(mutex);
if (is_wrt == 0)
  readcount++;
  if (readcount == 1)
    wait(wrt);

  signal(mutex);
  …
  reading is performed
  …
  wait(mutex);
    readcount--;
      if (readcount == 0)
        signal(wrt);
  signal(mutex);
else
  wait(rea)
```

버퍼에 접근하는 Writer 숫자 변수를 이용하여 writer이 버퍼에 접근하려고 하면 reader를 wait시킵니다.

웹검색을 통해 살펴본 자료들과는 기본적인 philosophy는 같으나, 구현의 방법이 크게 차이가 납니다.

찾아낸 하나의 예시를 아래에 첨부하여 봅니다.

```
class ReadWrite {
  … // as before
  private protected
  int waitingW = 0; // no of waiting Writers.

synchronized public void acquireRead() {
  while (writing || waitingW>0) {
    … wait(); …
```

```
  }
  ++readers;
}

synchronized public void releaseRead() {
  …
}

synchronized public void acquireWrite() {
  while (readers>0 || writing) {
    ++waitingW;
    …
    try{
      wait();
      …
      --waitingW;
    }
  writing = true;
}

synchronized public void releaseWrite() {… }
}
```

6.4 수업시간 과제

page out이 될 것 같습니다.

그 이유는 Data는 global 초깃값 변수인데, 이것은 읽고 쓰는 것이 가능한 변수입니다.

그 값이 변화하기에 프로그램으로부터 온전히 다시 가져오는 것이 불가능합니다.

따라서 page out이 필요할 듯 합니다.