M3239.005400 데이터사이언스를 위한 컴퓨팅 2 (001)

Homework #1

Due : 2024/09/29 (Sun)

2024-28413 이정현

1. Compilation Process

1.1 Preprocessing

(a)

	Path	Number of Lines
stdio.h	/usr/include/stdio.h	875
math.h	/usr/include/math.h	1341

(b) The option is **-E** / [gcc -E sqrt.c > sqrt_preprocessed.c]

```
1356
1357    extern int scanf (const char *__restrict __format, ...) ;
1358
```

```
1369                                    ;
1370    extern int scanf (const char *__restrict __format, ...) __asm__ ("" "__isoc99_scanf")
1371                                    ;
```

[scanf] in the preprocessed result.

*The reason for 2 declarations of [scanf].
The first picture is the basic declaration of [scanf].
The second picture is the declaration of the [scanf] function that follows the C99 standard. It indicates that the function needs to be assembled under the name “__isoc99_scanf”. (A safer version)

When the function is called by the program, the compiler will choose the appropriate version between the two declarations based on the system configuration.

In most current computer systems, the second declaration of the [scanf] function (C99 version) will be chosen by the compiler.

```
1308
1309    extern int printf (const char *__restrict __format, ...);
1310
```

[printf] in the preprocessed result

[sqrt] in the preprocessed result

(c) No the actual implementations are not included in the preprocessed results.

In the preprocessing stage, the contents of the header are extended and the declarations of the functions gets included. The actual implementations of the functions (scanf, printf, sqrt) are in external libraries which will be connected in the **linking** stage afterwards.

1.2 Compilation

(a) gcc -c sqrt.c -o sqrt.o

(b) The file format of [sqrt.o] is ELF (Executable and Linkable Format)

```
shpc106@elogin3:~/skeleton/hw1/sqrt$ file sqrt.o
sqrt.o: ELF 64-bit LSB relocatable, x86-64, version 1 (SYSV), not stripped
```

To find out the file format, we can use the [file] command. To be more precise about the [sqrt.o] file, it is a ELF file using 64 bits, with LSB (Least Significant Bit first) ordering.

1.3 Linking

(a) The error undefined reference to ‘sqrt’ means that the linker cannot find the definition of the function ‘sqrt’. This function is located in the math library(libm) in the C standard library. But gcc alone does not provide a link to this library as default. (This is related to reducing the size of a program)

Although the file [sqrt.o] is a compiled file, it does not contain actual implementations of external libraries. Therefore, we need to **explicitly link** the math library when we compile a code using a math function.

The correct code for creating a final executable [sqrt] file is shown below.

```
shpc106@elogin3:~/skeleton/hw1/sqrt$ gcc sqrt.o -lm -o sqrt
```

[gcc sqrt.o -lm -o sqrt]

(b) The following are the results of executing the [sqrt] program. Two numbers were tested, 5 and 16.

```
shpc106@elogin3:~/skeleton/hw1/sqrt$ ./sqrt 5
2.23606798
shpc106@elogin3:~/skeleton/hw1/sqrt$ ./sqrt 16
4.00000000
```

2. C Programming

2.1 Shift

(a) 1111 1111 1111 1111 1111 1111 1111 0000

(b) 1111 1111 1111 1111 1111 1111 1111 1100

(c) 0011 1111 1111 1111 1111 1111 1111 1100

(d)

Arithmetic Shift : Always preserves the sign bit. For negative numbers, the left side is filled with 1s. Therefore, it is mostly used on signed integers

Logical Shift : Always fills the left side with 0s. The sign bit is not preserved.

3. Cluster Practice

(a) Result of [sinfo]

```
shpc106@elogin3:~$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
class1       up       5:00      1    mix a02
class1       up       5:00      2  alloc a[00-01]
class1       up       5:00      9   idle a[03-11]
```

어떤 노드를 사용중이고, 사용할 수 있는지에 대한 정보를 알려주는 명령어이다.

사용가능한 partition과 상태, 그리고 각 partition의 시간 제한, 노드들의 상태를 알 수 있다.

(b) Result of [squeue]

```
shpc106@elogin3:~$ squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
```

유저들이 각각 slurm을 통해서 계산 노드에 접속하게 되는데, 어떤 유저가 얼만큼의 작업을 계산노드에 요청(할당) 했는지를 확인할 수 있는 명령어이다.

(c) Result of [srun -N 2 hostname]

```
shpc106@elogin3:~$ srun -N 2 hostname
a03
a04
```

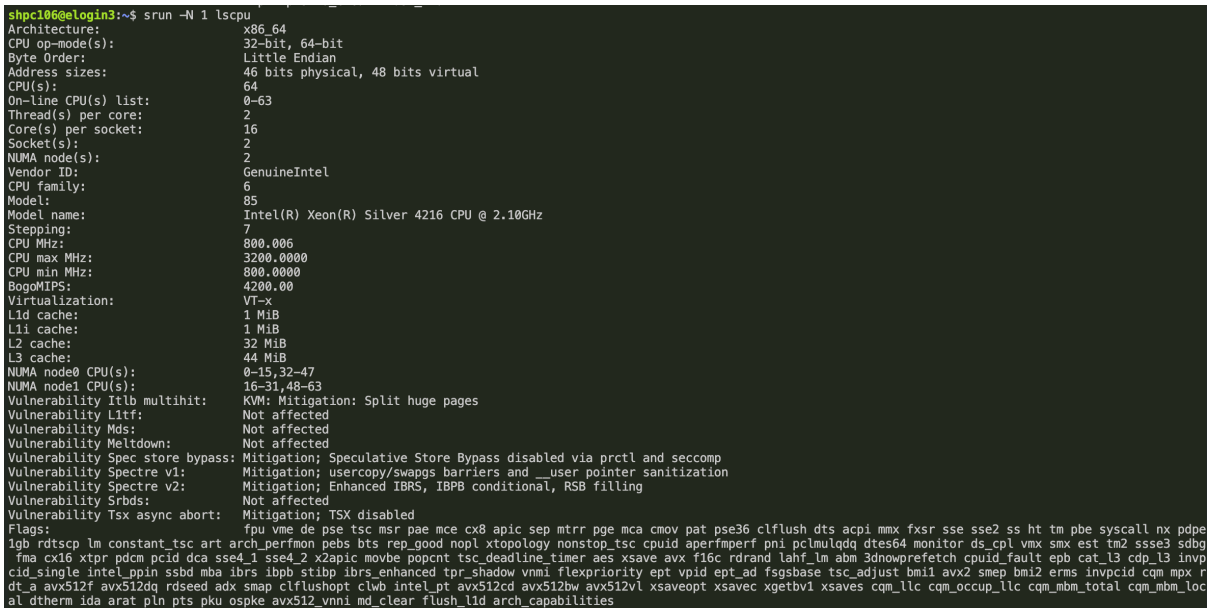
srun은 slurm에 제출하는 명령어이다. -N 2는 2개의 계산노드를 할당받아서 프로그램을 돌리고 싶다는 것을 의미한다. Hostname 자리에는 실제로 계산노드들이 실행했으면 하는 프로그램을 적는다. Hostname은 그냥 host의 이름을 반환한다. a03, a04는 위 명령으로 할당받은 두개의 계산 노드들 이다.

(d)

Result of [lscpu]

```
shpc106@elogin3:~$ lscpu
Architecture:                    x86_64
CPU op-mode(s):                  32-bit, 64-bit
Byte Order:                      Little Endian
Address sizes:                   46 bits physical, 48 bits virtual
CPU(s):                          32
On-line CPU(s) list:             0-31
Thread(s) per core:              2
Core(s) per socket:              8
Socket(s):                       2
NUMA node(s):                    2
Vendor ID:                       GenuineIntel
CPU family:                      6
Model:                           79
Model name:                      Intel(R) Xeon(R) CPU E5-2620 v4 @ 2.10GHz
Stepping:                        1
CPU MHz:                         1200.514
CPU max MHz:                     3000.0000
CPU min MHz:                     1200.0000
BogoMIPS:                        4199.96
Virtualization:                  VT-x
L1d cache:                       512 KiB
L1i cache:                       512 KiB
L2 cache:                        4 MiB
L3 cache:                        40 MiB
NUMA node0 CPU(s):               0-7,16-23
NUMA node1 CPU(s):               8-15,24-31
Vulnerability Itlb multihit:     KVM: Mitigation: Split huge pages
Vulnerability L1tf:              Mitigation; PTE Inversion; VMX conditional cache flushes, SMT vulnerable
Vulnerability Mds:               Mitigation; Clear CPU buffers; SMT vulnerable
Vulnerability Meltdown:          Mitigation; PTI
Vulnerability Spec store bypass: Mitigation; Speculative Store Bypass disabled via prctl and seccomp
Vulnerability Spectre v1:        Mitigation; usercopy/swapgs barriers and __user pointer sanitization
Vulnerability Spectre v2:        Mitigation; Full generic retpoline, IBPB conditional, IBRS_FW, STIBP conditional, RSB filli
                                 ng
Vulnerability Srbds:             Not affected
Vulnerability Tsx async abort:   Mitigation; Clear CPU buffers; SMT vulnerable
Flags:                           fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse36 clflush dts acpi mm
                                 x fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc arch_perfmon pebs bt
                                 s rep_good nopl xtopology nonstop_tsc cpuid aperfmperf pni pclmulqdq dtes64 monitor ds_cpl
                                 vmx smx est tm2 ssse3 sdbg fma cx16 xtpr pdcm pcid dca sse4_1 sse4_2 x2apic movbe popcnt ts
                                 c_deadline_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch cpuid_fault epb cat_l3
                                  cdp_l3 invpcid_single pti intel_ppin ssbd ibrs ibpb stibp tpr_shadow vnmi flexpriority ept
                                  vpid ept_ad fsgsbase tsc_adjust bmi1 hle avx2 smep bmi2 erms invpcid rtm cqm rdt_a rdseed
                                 adx smap intel_pt xsaveopt cqm_llc cqm_occup_llc cqm_mbm_total cqm_mbm_local dtherm ida ara
                                 t pln pts md_clear flush_l1d
```

Result of [srun -N 1 lscpu]



[lscpu] 명령어는 사용하고 있는 로컬 컴퓨터에 대한 CPU 구조 정보를 출력해준다.

[srun -N 1 lscpu] 명령어는 클러스터 안에 있는 계산노드들 중 하나를 srun으로 할당 받은 후, 그 계산노드에 대해 lscpu를 실행한다.

따라서 출력이 다른 이유는 전자는 현재 사용자가 사용하고 있는 로컬 시스템(로그인 노드나 로컬 워크스테이션)에 대한 정보를 출력해주는 것이고, 후자는 srun으로 먼저 클러스터의 계산 노드를 하나 할당받은 후, 그 계산 노드에 대한 정보를 출력해주는 것이다. 계산노드의 스펙은 로컬 시스템과 차이가 있을 수 있으므로 출력 결과가 다르게 나온다.