과제 #1

M1522.006700 확장형 고성능 컴퓨팅 (001)
M3239.005400 데이터사이언스를 위한 컴퓨팅 2 (001)

박찬정
서울대학교 전기정보공학부
2023-24013

1 (40점) Compilation Process

1.1 Preprocessing

(a) `cpp -v /dev/null` 명령어를 수행한 결과, 다음과 같은 출력을 얻을 수 있었다.

```
#include "..." search starts here:
#include <...> search starts here:
/usr/lib/gcc/x86_64-linux-gnu/9/include
/usr/local/include
/usr/include/x86_64-linux-gnu
/usr/include
```

이 정보를 바탕으로 첫 번째 디렉토리에서 네 번째 디렉토리까지 `find` 명령어를 사용하여 `stdio.h`, `math.h` 파일을 검색했다. `stdio.h` 파일의 경우 세 번째 디렉토리인 `/usr/include/x86_64-linux-gnu` 에서 `find` 명령어가 해당 파일을 찾아내었다.

```
$ find /usr/include/x86_64-linux-gnu -name "stdio.h"
/usr/include/x86_64-linux-gnu/bits/stdio.h
```

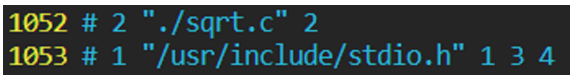
하지만 찾던 디렉토리에 바로 존재하지 않고, `bits` 폴더로 한번 더 들어가야 `stdio.h` 파일이 존재한다. 따라서 `cpp`는 이 파일을 찾지 못하고, 사용하지 않는다.

네 번째 디렉토리인 `/usr/include` 에서는 디렉토리 바로 아래의 두 파일을 찾을 수 있었다.

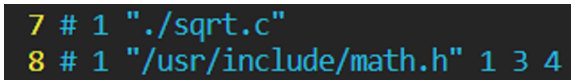
```
$ find /usr/include -name "stdio.h"
/usr/include/stdio.h
/usr/include/x86_64-linux-gnu/bits/stdio.h
/usr/include/c++/9/tr1/stdio.h

$ find /usr/include -name "math.h"
/usr/include/ucs/sys/math.h
/usr/include/math.h
/usr/include/c++/9/tr1/math.h
/usr/include/c++/9/math.h
```

즉 컴파일러는 각각 `/usr/include/stdio.h`, `/usr/include/math.h` 두 파일을 사용한다. 이는 cpp를 통해 실제 소스코드로 대체된 코드를 직접 살펴보아도 확인할 수 있다[Fig. 1]. 각 파일의 라인 수는 875, 1341 이다[Fig. 2].



(a) cpp 가 /usr/include/stdio.h를 사용한 모습.



(b) cpp 가 /usr/include/math.h를 사용한 모습.

Figure 1: cpp의 preprocess 결과를 통한 검증.

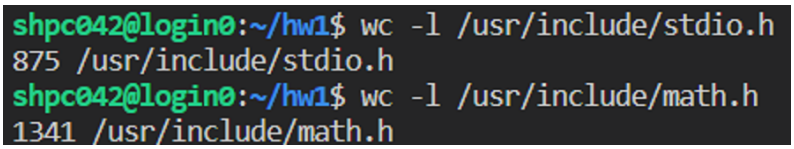


Figure 2: /usr/include/stdio.h, /usr/include/math.h 파일의 라인 수를 출력한 모습.

(b) Preprocess까지만 진행하는 gcc 옵션은 -E 이다. 출력 결과 중 scanf, printf, sqrt를 Fig. 3와 같이 찾을 수 있었다.

(c) 실제 구현이 들어있지 않다. sqrt.c 파일에서 참조한 파일들은 헤더 파일(.h)로, 보통 함수를 선언하는 코드로 채운다. 같은 헤더 파일을 공유함으로서 같은 심볼에 대해 같은 선언을 기반으로 컴파일할 수 있게 하기 위함이다. 실제 구현은 별도의 오브젝트 파일(.o) 또는 공유 오브젝트 파일(.so) 속에 준비되어 있으며, 링킹 과정에서 합쳐진다.

1.2 Compilation

(a) Object file을 출력하는 gcc 옵션은 -c 이다. 다음과 같은 명령어를 사용할 수 있다.

```
$ gcc -c sqrt.c
```

(b) Linux 운영체제에서는 파일을 다음과 같이 총 7가지로 분류한다.

- Regular files
- Directories
- Character device files
- Block device files
- Local domain sockets
- Named pipes (FIFOs)
- Symbolic links

파일이 이 중 어디에 속하는지는 stat 명령어를 사용하여 알아낼 수 있다.

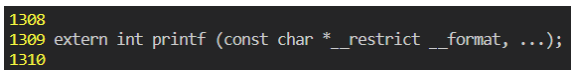
```
$ stat -c "%F" sqrt.o
regular file
```

즉 sqrt.o 파일은 Linux 운영체제의 파일 분류 중 regular file에 해당한다.

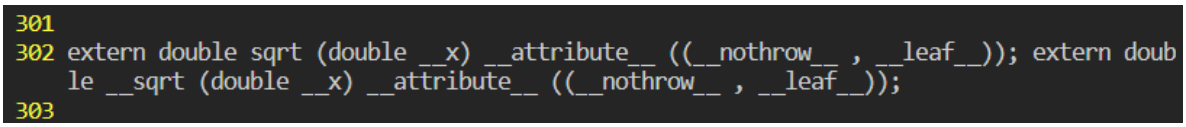
추가로, Linux 파일에서는 magic number라는 시스템을 사용한다. 이는 파일의 최초 2바이트 영역에 저장되는 숫자를 말하며, 파일의 타입에 대한 간단한 메타데이터를 제공하는 역할을 한다. file 명령어를 사용하여 파일의 magic number로부터 파일에 대한 정보를 얻을 수 있다.

```
$ file sqrt.o
sqrt.o: ELF 64-bit LSB relocatable, x86-64, version 1 (SYSV), not stripped
```

(a) `scanf`



(b) printf



(c) sqrt

Figure 3: Preprocess된 출력 결과 중 `scanf`, `printf`, `sqrt` 에 대한 부분

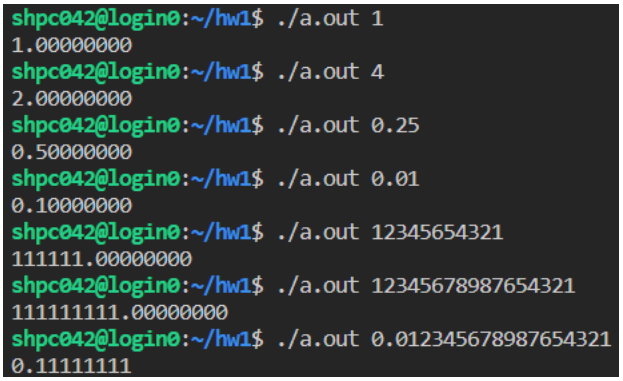


Figure 4: `sqrt`에 임의의 수를 입력한 결과.

1.3 Linking

(a) `sqrt` 함수에 대한 구현 부분이 컴파일 과정에 제공되지 않았기 때문에 링킹 과정에서 에러가 발생하였다. 올바르게 컴파일하기 위해서는 `sqrt` 함수에 대한 구현이 포함된 오브젝트 파일을 `-lm` 명령어를 통해 제공해야 한다. 결과적으로 다음과 같은 명령어를 통해 올바르게 컴파일할 수 있다.

```
$ gcc sqrt.o -lm
```

(b) Fig. 4과 같이 `sqrt` 함수를 실행해보았다.

2 (40점) Compilation Process

`convert.c` 파일을 작성하여 제출하였다.

3 (20점) 클러스터 사용 연습

(a) Fig. 5와 같이 `sinfo` 명령어를 실행해보았다. 이 명령어는 Slurm 시스템의 node와 partition에 대한 정보를 제공한다. 출력의 각 열은 다음과 같은 의미를 갖는다.

- **PARTITION**: Partition의 이름.
- **AVAIL**: 해당 partition의 사용 가능 여부.
- **TIMELIMIT**: 하나의 job이 해당 partition을 사용할 수 있는 최대 시간.
- **NODES**: 해당 partition에 할당된 node의 수.
- **STATE**: 해당 partition의 상태(job 수행 여부).
- **NODELIST**: 해당 partition에 할당된 노드의 이름의 목록.

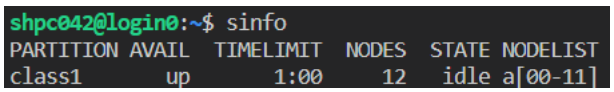


Figure 5: `sinfo` 명령어를 수행한 결과.

Figure 6: `squeue` 명령어를 수행한 결과.

(b) Fig. 6와 같이 `squeue` 명령어를 실행해보았다. 이 명령어는 Slurm 시스템의 작업 대기열에 있는 작업들에 대한 정보를 제공한다. 출력의 각 열은 다음과 같은 의미를 갖는다.

- **JOBID**: 작업의 ID. 각 작업마다 고유한 ID를 갖는다.
- **PARTITION**: 작업이 실행중인 partition.
- **NAME**: 작업의 이름.
- **USER**: 작업을 생성한 사용자의 ID
- **ST**: 작업의 현재 상태.
- **TIME**: 작업이 partition을 사용한 시간.
- **NODES**: 작업에 할당된 노드의 개수.
- **NODELIST**: 작업에 할당된 노드의 이름의 목록.

(c) Fig. 7와 같이 `srun -p class1 -N 2 hostname` 명령어를 실행해보았다. 이 명령어는 `class1` partition에서 2개의 노드를 할당받아 `hostname` 명령어를 수행한다. `hostname` 명령어는 Linux 시스템 사용자의 이름을 출력하는 명령어이다. 따라서 두 줄의 출력은 각각 할당받은 두 노드의 사용자 이름이다.

(d) Fig. 8와 같이 `lscpu` 명령어를 실행해보았다. 이 명령어는 현재 시스템의 CPU에 대한 정보를 제공한다. 로그인 노드에서 실행했으므로 로그인 노드의 CPU 정보를 출력한다. 몇 가지 출력만 살펴보면 프로세서는 AMD EPYC 7502 32-Core Processor이며 L1d, L1i 캐시는 각각 2 MiB, L2 캐시는 32 MiB, L3 캐시는 256 MiB이다.

또한 Fig. 9와 같이 `srun -p class1 -N 1 lscpu` 명령어를 실행해보았다. 이 명령어는 `class1` partition에서 1개의 노드를 할당받아 `lscpu` 명령어를 수행한다. 이 경우 컴퓨트 노드에서 실행하게 되므로 컴퓨트 노드의 CPU 정보를 출력한다. 몇 가지 출력만 살펴보면 프로세서는 Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz이며 L1d, L1i 캐시는 각각 1 MiB, L2 캐시는 32 MiB, L3 캐시는 44 MiB 이다.

두 명령의 출력이 다른 이유는 각자 다른 환경(로그인 노드, 컴퓨트 노드)에서 실행되었기 때문에 각자의 CPU에 대한 정보를 출력하기 때문이다.

```
shpc042@login0:~$  srun -p class1 -N 2 hostname
a00
a01
```

Figure 7: `srun` 명령어를 수행한 결과.

```
shpc042@login0:~$ lscpu
Architecture:                    x86_64
CPU op-mode(s):                  32-bit, 64-bit
Byte Order:                      Little Endian
Address sizes:                   43 bits physical, 48 bits virtual
CPU(s):                          128
On-line CPU(s) list:             0-127
Thread(s) per core:              2
Core(s) per socket:              32
Socket(s):                       2
NUMA node(s):                    2
Vendor ID:                       AuthenticAMD
CPU family:                      23
Model:                           49
Model name:                      AMD EPYC 7502 32-Core Processor
Stepping:                        0
Frequency boost:                 enabled
CPU MHz:                         1896.434
CPU max MHz:                     2500.0000
CPU min MHz:                     1500.0000
BogoMIPS:                        4999.74
Virtualization:                  AMD-V
L1d cache:                       2 MiB
L1i cache:                       2 MiB
L2 cache:                        32 MiB
L3 cache:                        256 MiB
NUMA node0 CPU(s):               0-31,64-95
NUMA node1 CPU(s):               32-63,96-127
Vulnerability Itlb multihit:     Not affected
Vulnerability L1tf:              Not affected
Vulnerability Mds:               Not affected
Vulnerability Meltdown:          Not affected
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 AMD retpoline, IBPB conditional, IBRS_FW, STIBP cond
                                 itional, RSB filling
Vulnerability Srbds:             Not affected
Vulnerability Tsx async abort:   Not affected
Flags:                           fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse
                                 36 clflush mmx fxsr sse sse2 ht syscall nx mmxext fxsr_opt pdpe1gb rd
                                 tscp lm constant_tsc rep_good nopl nonstop_tsc cpuid extd_apicid aper
                                 fmperf pni pclmulqdq monitor ssse3 fma cx16 sse4_1 sse4_2 movbe popcn
                                 t aes xsave avx f16c rdrand lahf_lm cmp_legacy svm extapic cr8_legacy
                                  abm sse4a misalignsse 3dnowprefetch osvw ibs skinit wdt tce topoext
                                 perfctr_core perfctr_nb bpext perfctr_llc mwaitx cpb cat_l3 cdp_l3 hw
                                 _pstate sme ssbd mba sev ibrs ibpb stibp vmmcall fsgsbase bmi1 avx2 s
                                 mep bmi2 cqm rdt_a rdseed adx smap clflushopt clwb sha_ni xsaveopt xs
                                 avec xgetbv1 xsaves cqm_llc cqm_occup_llc cqm_mbm_total cqm_mbm_local
                                  clzero irperf xsaveerptr wbnoinvd arat npt lbrv svm_lock nrip_save t
                                 sc_scale vmcb_clean flushbyasid decodeassists pausefilter pfthreshold
                                  avic v vmsave vmload vgif umip rdpid overflow recov succor smca
```

Figure 8: `lscpu` 명령어를 수행한 결과.

```
shpc042@login0:~$ srun -p class1 -N 1 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):                          64
On-line CPU(s) list:             0-63
Thread(s) per core:              2
Core(s) per socket:              16
Socket(s):                       2
NUMA node(s):                    2
Vendor ID:                       GenuineIntel
CPU family:                      6
Model:                           85
Model name:                      Intel(R) Xeon(R) Silver 4216 CPU @ 2.10GHz
Stepping:                        7
CPU MHz:                         800.001
CPU max MHz:                     3200.0000
CPU min MHz:                     800.0000
BogoMIPS:                        4200.00
Virtualization:                  VT-x
L1d cache:                       1 MiB
L1i cache:                       1 MiB
L2 cache:                        32 MiB
L3 cache:                        44 MiB
NUMA node0 CPU(s):               0-15,32-47
NUMA node1 CPU(s):               16-31,48-63
Vulnerability Itlb multihit:     KVM: Mitigation: Split huge pages
Vulnerability L1tf:              Not affected
Vulnerability Mds:               Not affected
Vulnerability Meltdown:          Not affected
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; Enhanced IBRS, IBPB conditional, RSB filling
Vulnerability Srbds:             Not affected
Vulnerability Tsx async abort:   Mitigation; TSX disabled
Flags:                           fpu vme de pse tsc msr pae mce cx8 apic sep mtrr pge mca cmov pat pse
36 clflush dts acpi mmx fxsr sse sse2 ss ht tm pbe syscall nx pdpe1gb rdtscp lm constant_tsc art arch_
perfmon pebs bts 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 tsc_deadl
ine_timer aes xsave avx f16c rdrand lahf_lm abm 3dnowprefetch cpuid_fault epb cat_l3 cdp_l3 invpcid_si
ngle intel_ppin ssbd mba ibrs ibpb stibp ibrs_enhanced tpr_shadow vnmi flexpriority ept vpid ept_ad fs
gsbase tsc_adjust bmi1 avx2 smep bmi2 erms invpcid cqm mpx rdt_a avx512f avx512dq rdseed adx smap clfl
ushopt clwb intel_pt avx512cd avx512bw avx512vl xsaveopt xsavec xgetbv1 xsaves cqm_llc cqm_occup_llc c
qm_mbm_total cqm_mbm_local dtherm ida arat pln pts pku ospke avx512_vnni md_clear flush_l1d arch_capab
ilities
```

Figure 9: `srun -p class1 -N 1 lscpu` 명령어를 수행한 결과.