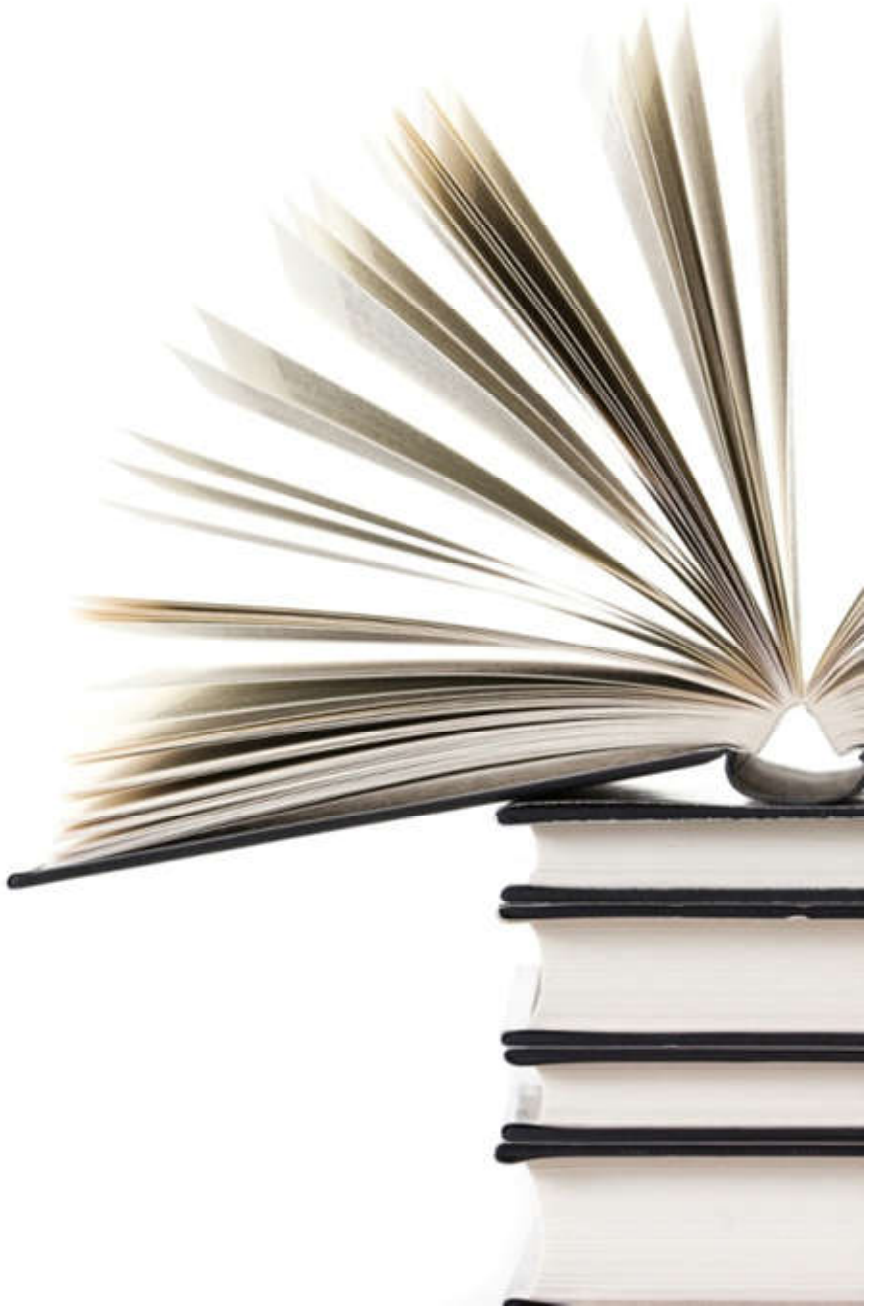
KiSTi SUPERCOMPUTING EDUCATION CENTER
www.kisti.re.kr

Linux 사용법



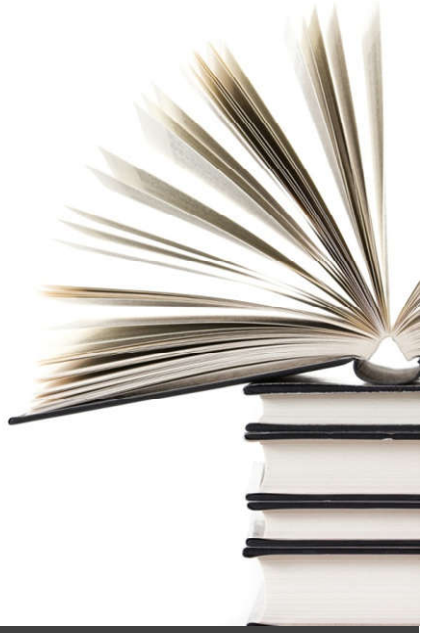
http://www.ksc.re.kr
http://webedu.ksc.re.kr

National Institute of Supercomputing and Networking 1967

Schedule

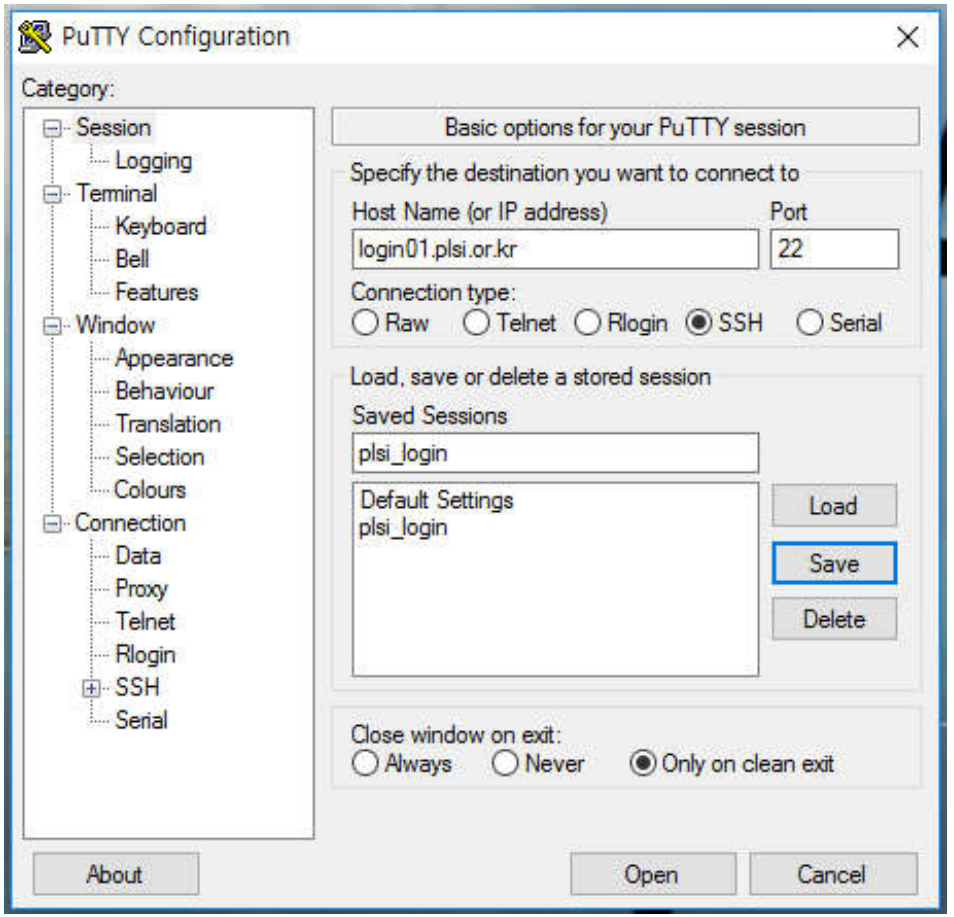
09:30	-	10:30	▪ 시스템 접속 및 Linux Command
10:30	-	10:40	▪ Break
10:40	-	12:00	▪ Vi editor
12:00	-	13:00	▪ Lunch
13:00	-	13:50	▪ OpenMP
13:50	-	14:00	▪ Break
14:00	-	14:50	▪ OpenMP
14:50	-	15:00	▪ Break
15:00	-	16:30	▪ OpenMP / LoadLeveler
16:30	-	17:00	▪ Summary

시스템 접속



Putty 설정

- **Host Name: login01.plsi.or.kr**
- **SSH Port: 22**
- **Save**

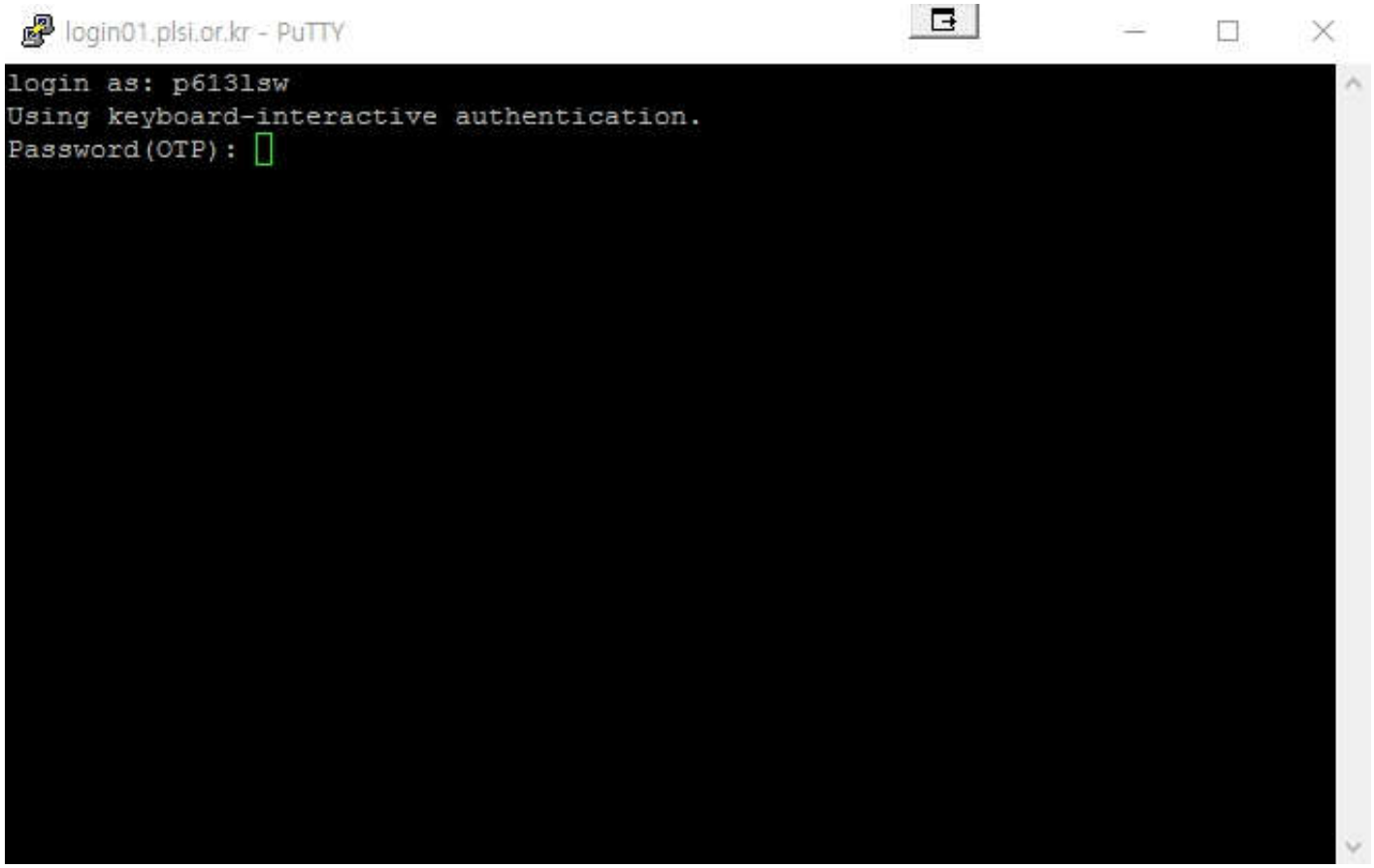


Putty 설정

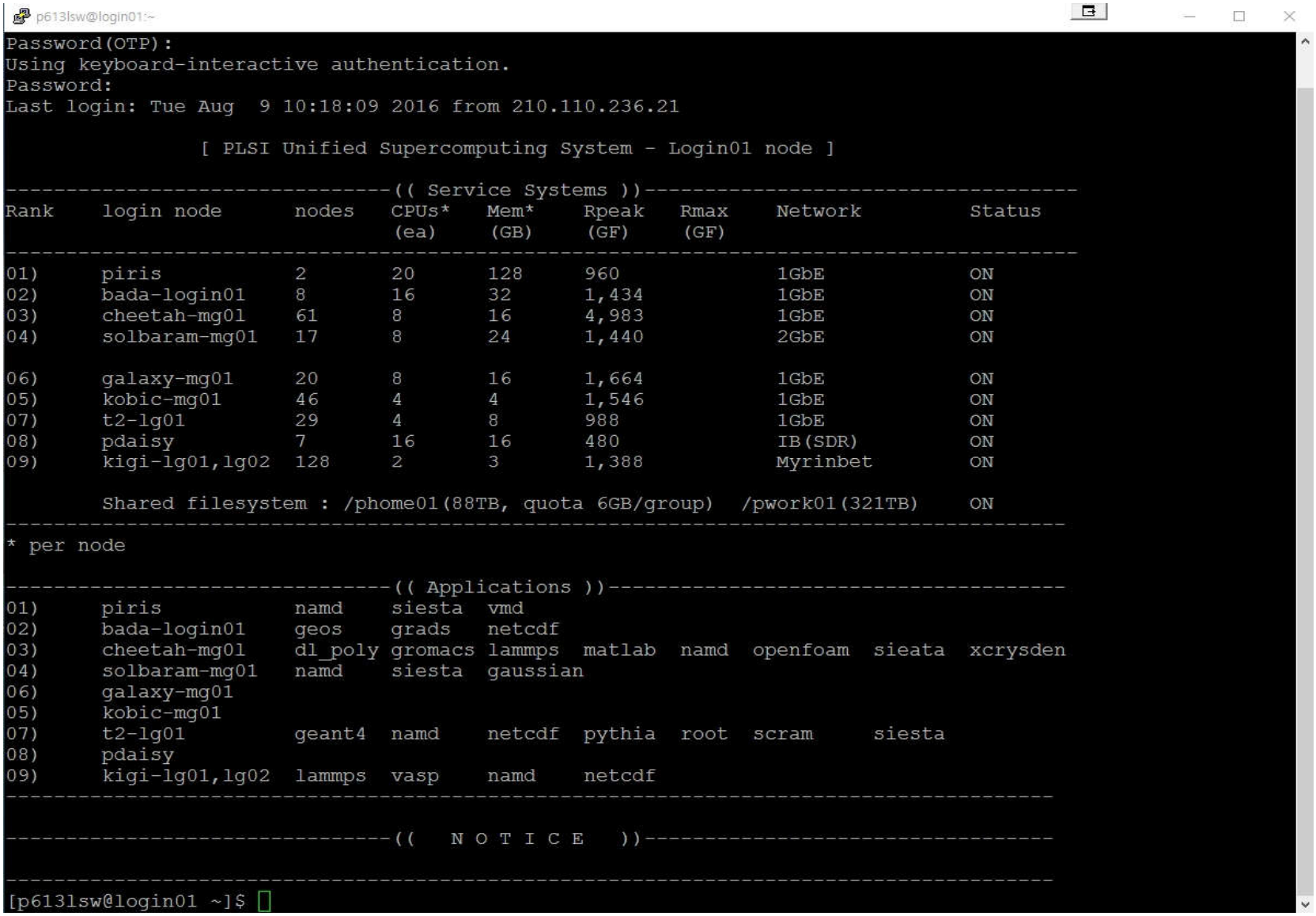
➢**PuTTY Security Alert [예(y)] 선택**

Putty 설정

- login as :
- Password(OTP) :
- Password

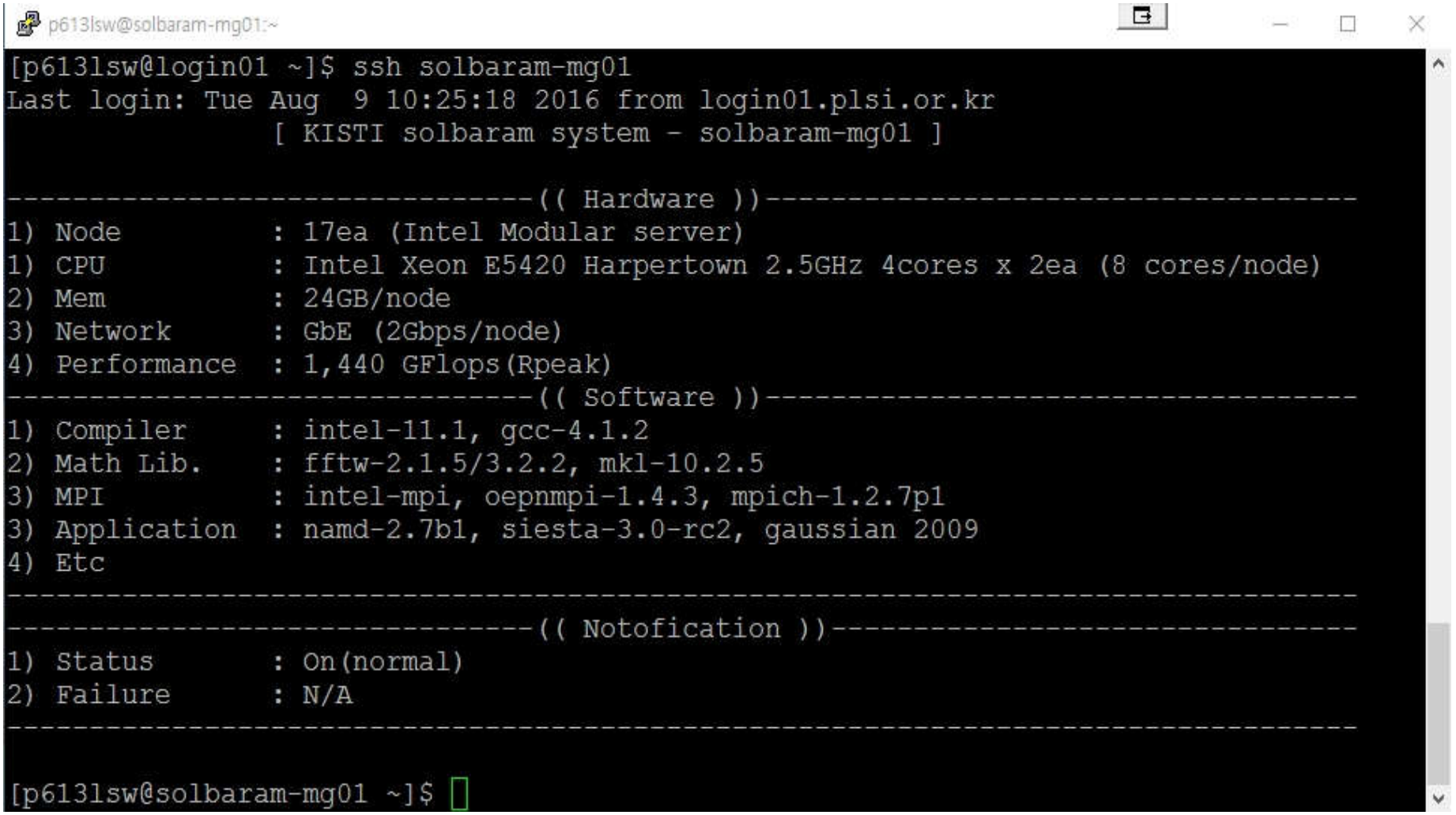


Putty 설정


p613lsw@login01:~
Password(OTP):
Using keyboard-interactive authentication.
Password:
Last login: Tue Aug 9 10:18:09 2016 from 210.110.236.21
[PLSI Unified Supercomputing System - Login01 node]
((Service Systems))
Rank login node nodes CPUs* (ea) Mem* (GB) Rpeak (GF) Rmax (GF) Network Status
01) piris 2 20 128 960 1GbE ON
02) bada-login01 8 16 32 1,434 1GbE ON
03) cheetah-mg01 61 8 16 4,983 1GbE ON
04) solbaram-mg01 17 8 24 1,440 2GbE ON
06) galaxy-mg01 20 8 16 1,664 1GbE ON
05) kobic-mg01 46 4 4 1,546 1GbE ON
07) t2-lg01 29 4 8 988 1GbE ON
08) pdaisy 7 16 16 480 IB(SDR) ON
09) kigi-lg01,lg02 128 2 3 1,388 Myrinbet ON
Shared filesystem : /phome01(88TB, quota 6GB/group) /pwork01(321TB) ON
* per node
((Applications))
01) piris namd siesta vmd
02) bada-login01 geos grads netcdf
03) cheetah-mg01 dl_poly gromacs lammps matlab namd openfoam sieata xcrysden
04) solbaram-mg01 namd siesta gaussian
06) galaxy-mg01
05) kobic-mg01
07) t2-lg01 geant4 namd netcdf pythia root scram siesta
08) pdaisy
09) kigi-lg01,lg02 lammps vasp namd netcdf
((N O T I C E))
[p613lsw@login01 ~]$

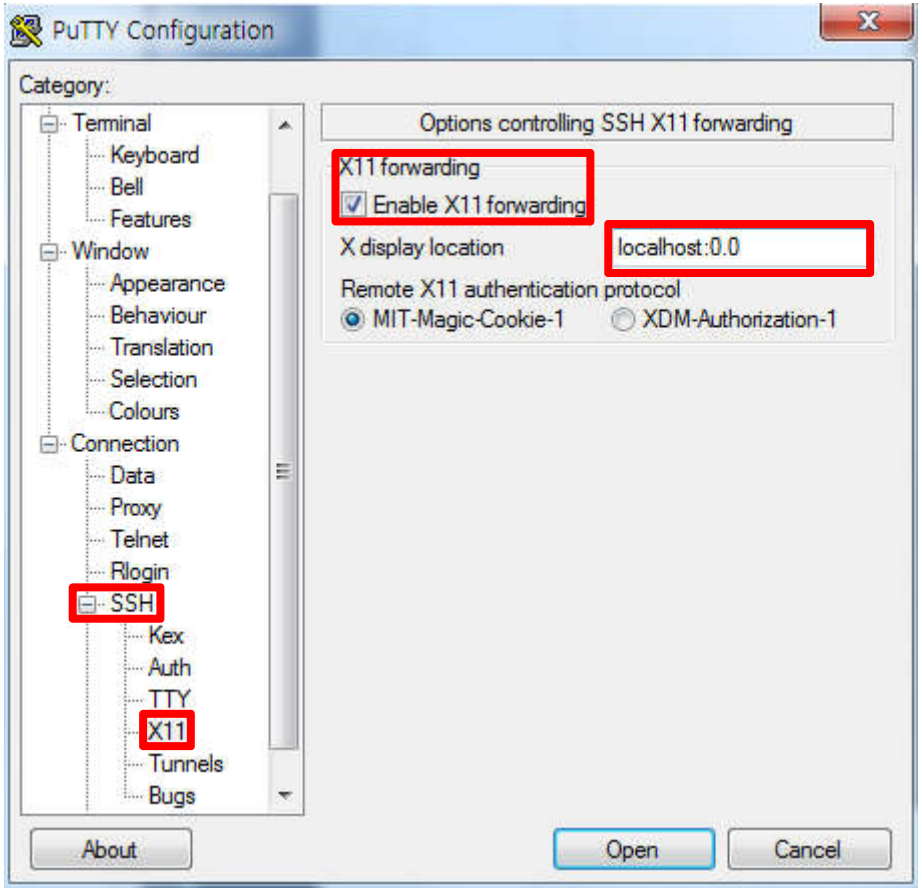
Putty 설정

➢ssh solbaram-mg01



Putty 설정

- ssh => X11 [Enable X11 forwarding] 체크
- X display location : localhost : 0.0
- save



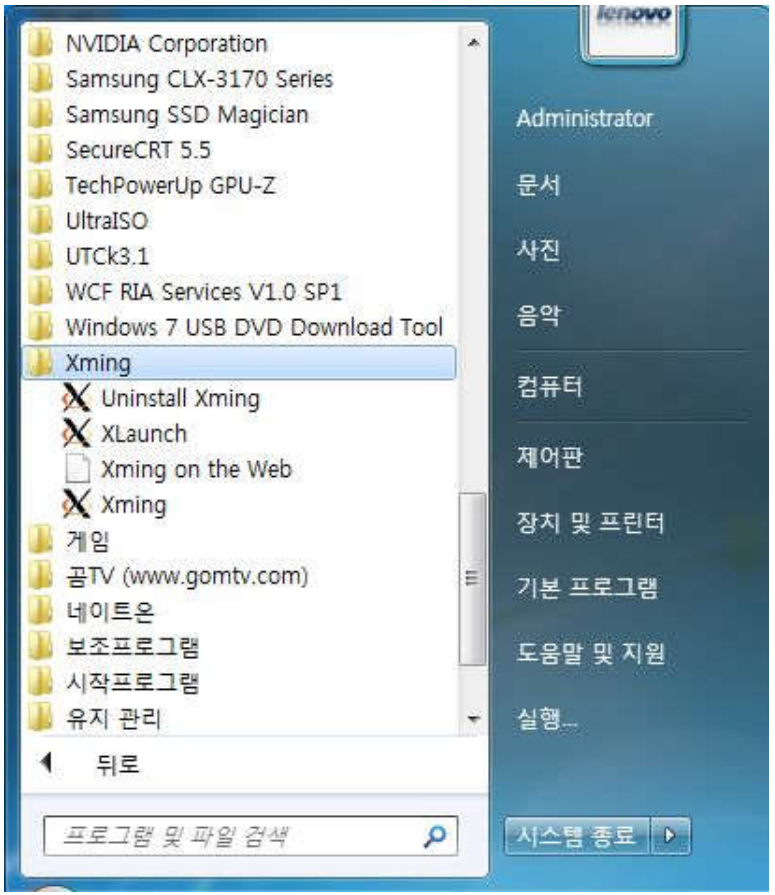
윈도우에서
Xming 실행

Xming

➢**Freeware**

➢다운로드 웹싸이트

- http://www.google.com 에서 Xming으로 검색
- http://sourceforge.net/projects/xming/



➢설치 및 사용방법

- 실행 파일 다운로드 후 설치
- Xming 실행
- Xwindows GUI 프로그램을 사용하기 위해서는 먼저 실행 필수

➢X client 실행

```
--------------------------------------------------------------------------------
/usr/bin/xauth:  creating new authority file /phome01/p613lsw/.Xauthority
[p613lsw@login01 ~]$ l
-bash: l: command not found
[p613lsw@login01 ~]$ l
-bash: l: command not found
[p613lsw@login01 ~]$ ssh solbaram-mg01
Last login: Tue Aug  9 10:27:18 2016 from login01.plsi.or.kr
               [ KISTI solbaram system - solbaram-mg01 ]

--------------------------------(( Hardware ))----------------------------------
1) Node         : 17ea (Intel Modular server)
1) CPU          : Intel Xeon E5420 Harpertown 2.5GHz 4cores x 2ea (8 cores/node)
2) Mem          : 24GB/node
3) Network      : GbE (2Gbps/node)
4) Performance  : 1,440 GFlops(Rpeak)
--------------------------------(( Software ))----------------------------------
1) Compiler     : intel-11.1, gcc-4.1.2
2) Math Lib.    : fftw-2.1.5/3.2.2, mkl-10.2.5
3) MPI          : intel-mpi, oepnmpi-1.4.3, mpich-1.2.7p1
3) Application  : namd-2.7b1, siesta-3.0-rc2, gaussian 2009
4) Etc
--------------------------------------------------------------------------------
-------------------------------(( Notofication ))-------------------------------
1) Status       : On(normal)
2) Failure      : N/A
--------------------------------------------------------------------------------

[p613lsw@solbaram-mg01 ~]$ xclock

[p613lsw@solbaram-mg01 ~]$ xeye
-bash: xeye: command not found
[p613lsw@solbaram-mg01 ~]$ xclock
```

Solbaram System

- 운영체제 : **CentOS 5.5 (LINUX, 64bit)**
- **CPU** 모델 / 성능 : **Intel Xeon Harpertown (E5420) / 2.50GHz**
- 노드 수 : **18**개 **node / 8 Core per 1 node**
- 메모리 : **24GB DDR3 Memory per 1 node**
- 총 **CORE** 수 : **144 Core**
- 로그인 노드명 : **solbaram-mg01**

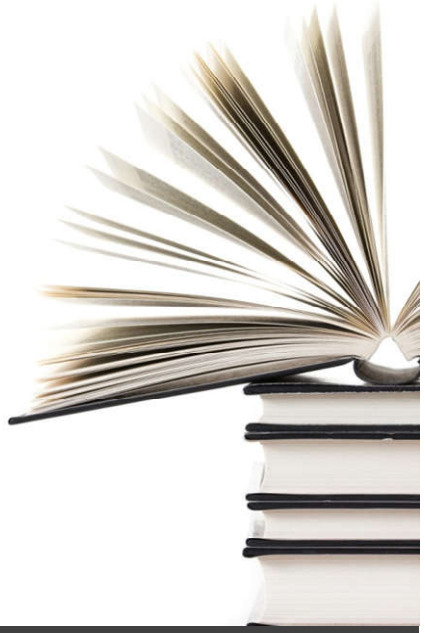
Checking Environment

➢Compiler Environment

- Use **module** command
- module av
- module add intel/intel-11.1 mpi/intel/openmpi-1.4.3
- module li

```
[p613lsw@solbaram-mg01 ~]$ module li
No Modulefiles Currently Loaded.
[p613lsw@solbaram-mg01 ~]$ module av

------------------------------------------------------------ /usr/share/Modules/modulefiles --------------
dot         module-cvs  module-info modules     null        use.own

------------------------------------------------------------------- /etc/modulefiles ---------------------
fftw/2.1.5/gcc/double/fftw-2.1.5                    fftw/3.3.4/mpich-3.2/intel-11.1/double/fftw-3.3.4
fftw/2.1.5/gcc/single/fftw-2.1.5                    gauss/g09.d01
fftw/2.1.5/intel-11.1/double/fftw-2.1.5             intel/intel-11.1
fftw/2.1.5/intel-11.1/single/fftw-2.1.5             intel/mkl-10.2.5
fftw/2.1.5/openmpi-1.4.3/gcc/double/fftw-2.1.5      mpi/gcc/mpich-1.2.7p1
fftw/2.1.5/openmpi-1.4.3/gcc/single/fftw-2.1.5      mpi/gcc/openmpi-1.4.3
fftw/2.1.5/openmpi-1.4.3/intel-11.1/double/fftw-2.1.5 mpi/intel/mpich-1.2.7p1
fftw/2.1.5/openmpi-1.4.3/intel-11.1/single/fftw-2.1.5 mpi/intel/mpich-3.2
fftw/3.2.2/gcc/double/fftw-3.2.2                    mpi/intel/openmpi-1.4.3
fftw/3.2.2/gcc/single/fftw-3.2.2                    namd/namd-2.7b1
fftw/3.2.2/intel-11.1/double/fftw-3.2.2             python/python-2.7.12
fftw/3.2.2/intel-11.1/single/fftw-3.2.2             siesta/siesta-3.0-rc2
[p613lsw@solbaram-mg01 ~]$ module add intel/intel-11.1 mpi/intel/openmpi-1.4.3
[p613lsw@solbaram-mg01 ~]$ module li
Currently Loaded Modulefiles:
  1) intel/mkl-10.2.5          2) intel/intel-11.1          3) mpi/intel/openmpi-1.4.3
[p613lsw@solbaram-mg01 ~]$
```

Structure

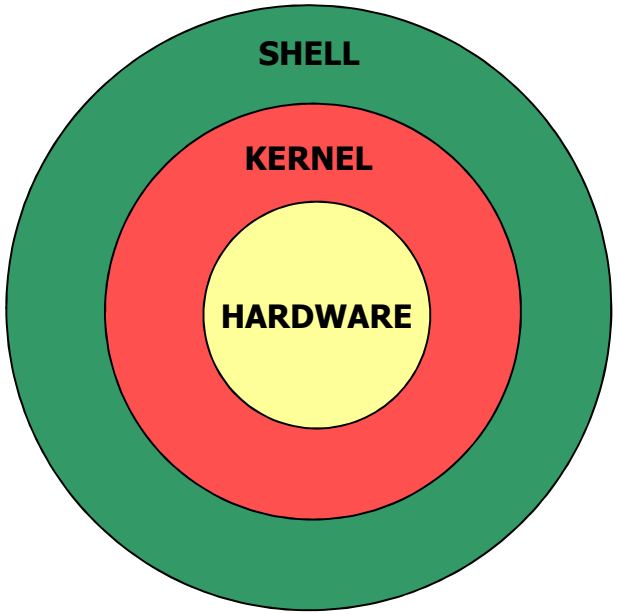


Structure

➢**Kernel**
- 운영체제의 핵심
- 시스템 자원들을 관리하는 부분

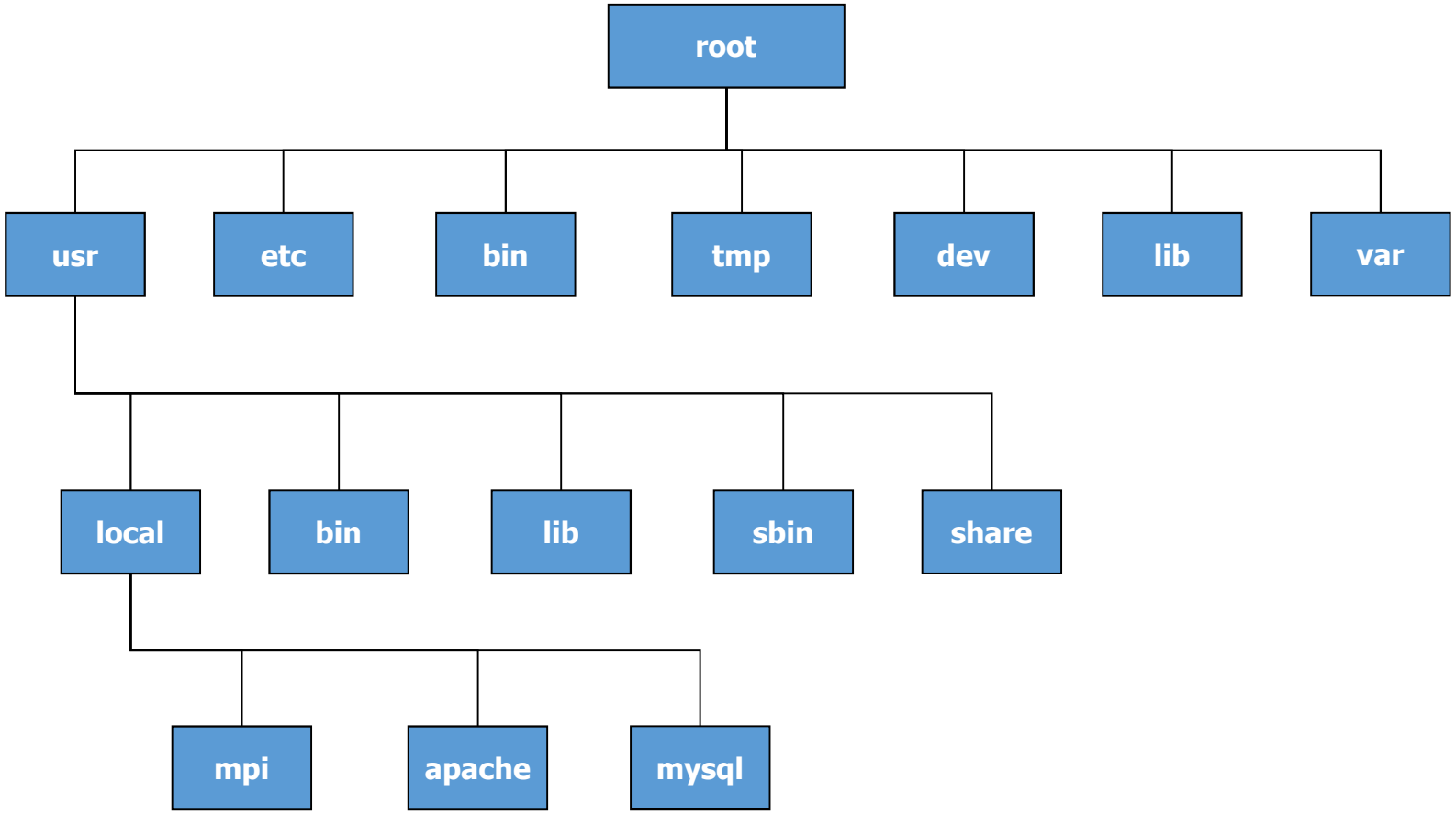
➢**Shell**
- 명령어 해석기
- 사용자와 시스템간의 명령의 전달

➢**File System**
- 계층구조를 이루고 있음
- 정보들을 저장할 수 있는 장소

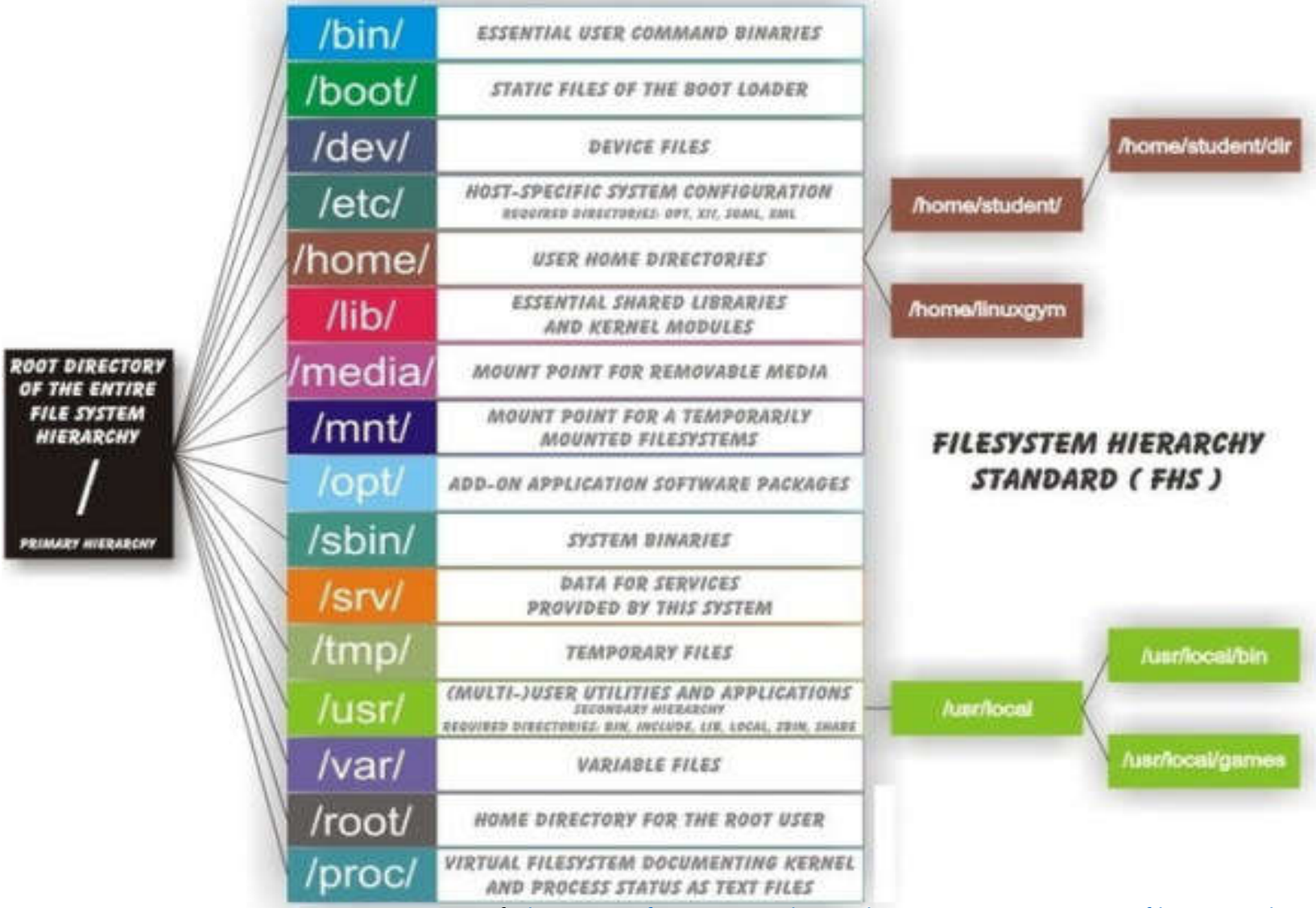


❖여러 유저가 하나의 시스템을 공유할 수 있도록 설계

File Hierarchy



File Hierarchy



ref. http://configureinstantly.wordpress.com/2011/01/23/filesystem-hierarchy-standard/

명령어 - 구조

➢명령어 조합

➢**Example**

```
$ ls
$ ls -a
$ ls -a /home
```

명령어 - 구조

➢ Command

- command
  - 일반적으로 소문자
  - 무엇을 할 것인지

- option
  - 불필요한 경우도 있음
  - 명령어 출력 향상
  - 사용자 필요에 맞추어 출력
  - 하나 이상의 다른 옵션과 함께 조합하여 사용

- argument
  - 사용자의 명령어에 따라 실행
  - 하나 이상의 argument 를 가질 수 있음
  - 불필요한 경우도 있음

First (and Last) Commands

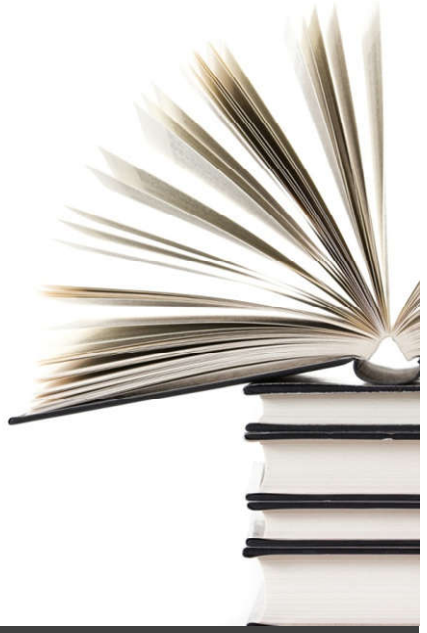
➢Login

- User ID
  - assigned by systems administrator
  - probably won't change
- Password
  - assigned by systems administrator
  - should not share it

➢Logoff

- exit
  - this is usually the way to log off
  - may differ from system to system

Basic Command



who, whoami

➢**who, whoami**

- 시스템에 Login 한 사용자 및 자신이 누구인지 보여준다
- $ who [am i]

```
$ who
edun30   pts/0        2012-11-01 09:36 (master01)
edun30   pts/1        2012-11-01 09:41 (s0003)
$
$ who am i
edun30   pts/1        2012-11-01 09:41 (s0003)
$
```

접속한
서버이름

ls

➢ ls

- 디렉터리 내의 파일 목록을 보기 위한 명령

옵 션	설 명
-l	파일 및 디렉토리를 list 형식으로 출력
-a	모든 파일, 디렉토리 출력 (숨김파일 포함)
-R	하위 디렉토리의 파일까지 보여주기
-S	파일 크기가 큰 순서로 출력
-F	파일뒤에 구분자 표시 (디렉토리는 /, 실행파일은 *)

ls

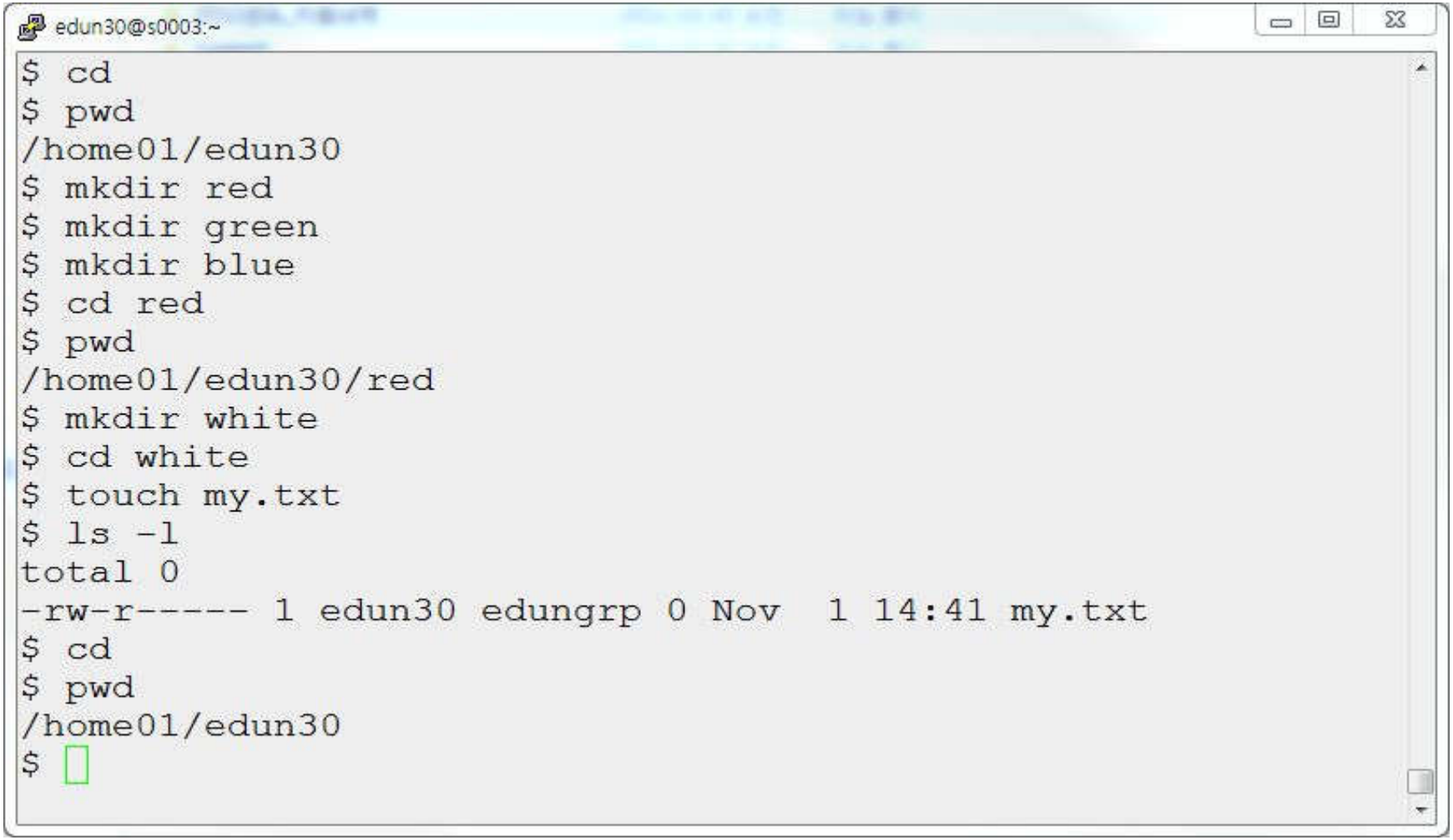
➢ls 명령어 기본 사용법

디렉터리 이동/생성/삭제 관련

- **cd**
  - 디렉터리 이동 명령
- **pwd**
  - 현재 디렉터리 위치를 보여줌
- **mkdir**
  - 새로운 디렉터리를 만들 때 사용
- **rmdir**
  - 디렉터리를 삭제할 때 사용
- **touch**
  - 파일이 존재할 경우 파일의 수정날짜를 변경, 존재하지 않을 경우 0kb 파일 생성
- **cp**
  - 파일 복사 명령
  - 속성을 유지할 경우 -a 옵션 추가

디렉터리 이동/생성/삭제 관련

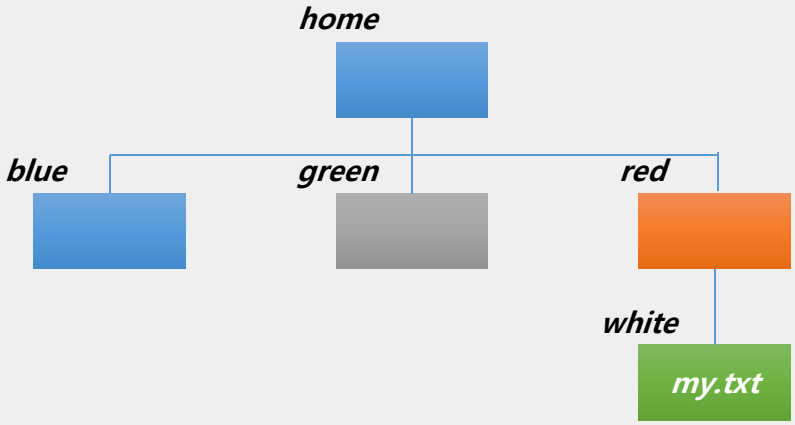
➢기본 사용법 **(cd, pwd, mkdir, touch)**



디렉터리 이동/생성/삭제 관련

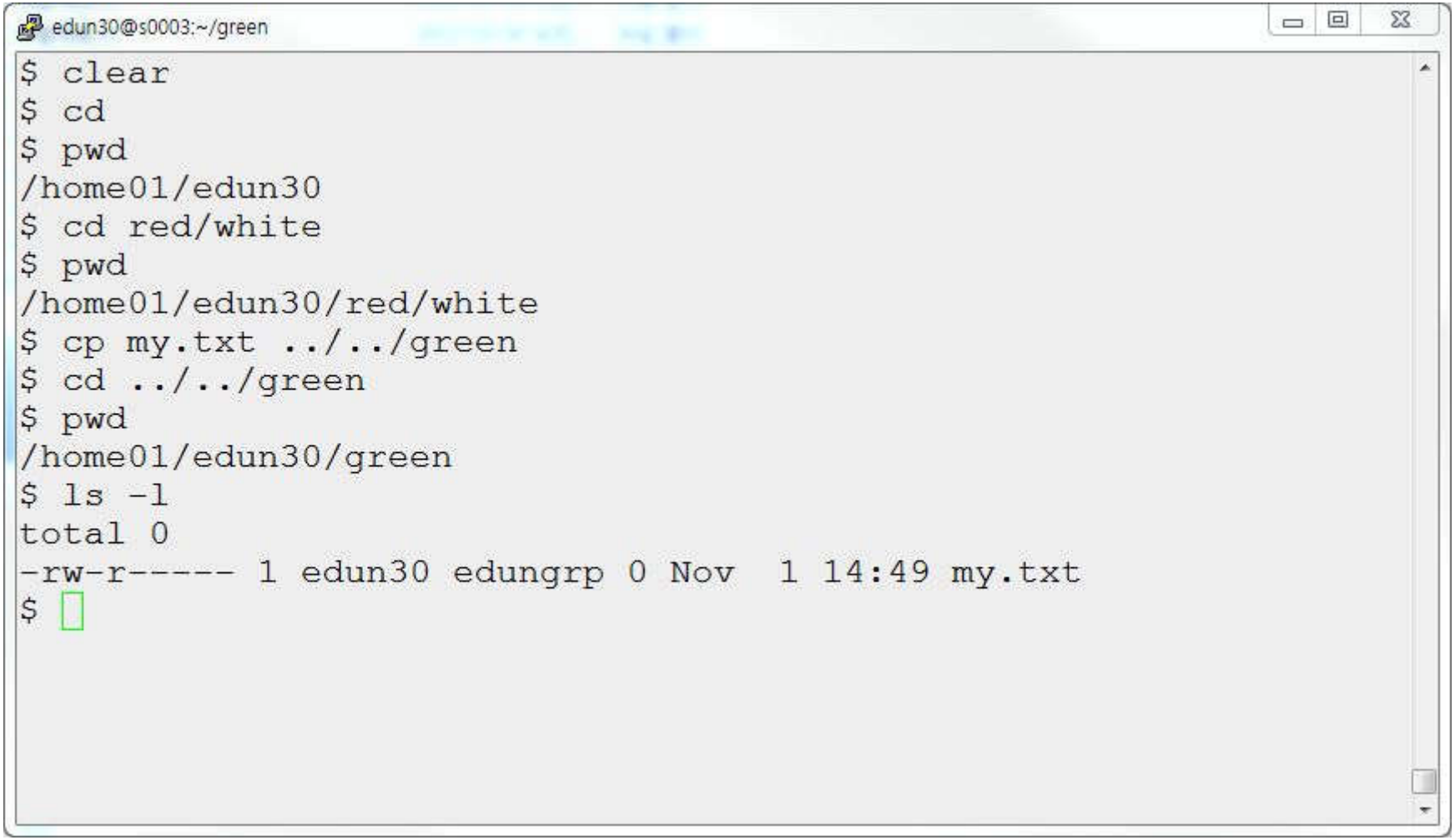
➢기본 사용법 **(tree)**

```
edun30@s0003:~
$ cd
$ pwd
/home01/edun30
$
$
$ tree
.
|-- blue
|-- green
`-- red
    `-- white
        `-- my.txt

4 directories, 1 file
$
```



디렉터리 이동/생성/삭제 관련

➢기본 사용법 **(cp, clear)**



디렉터리 이동/생성/삭제 관련

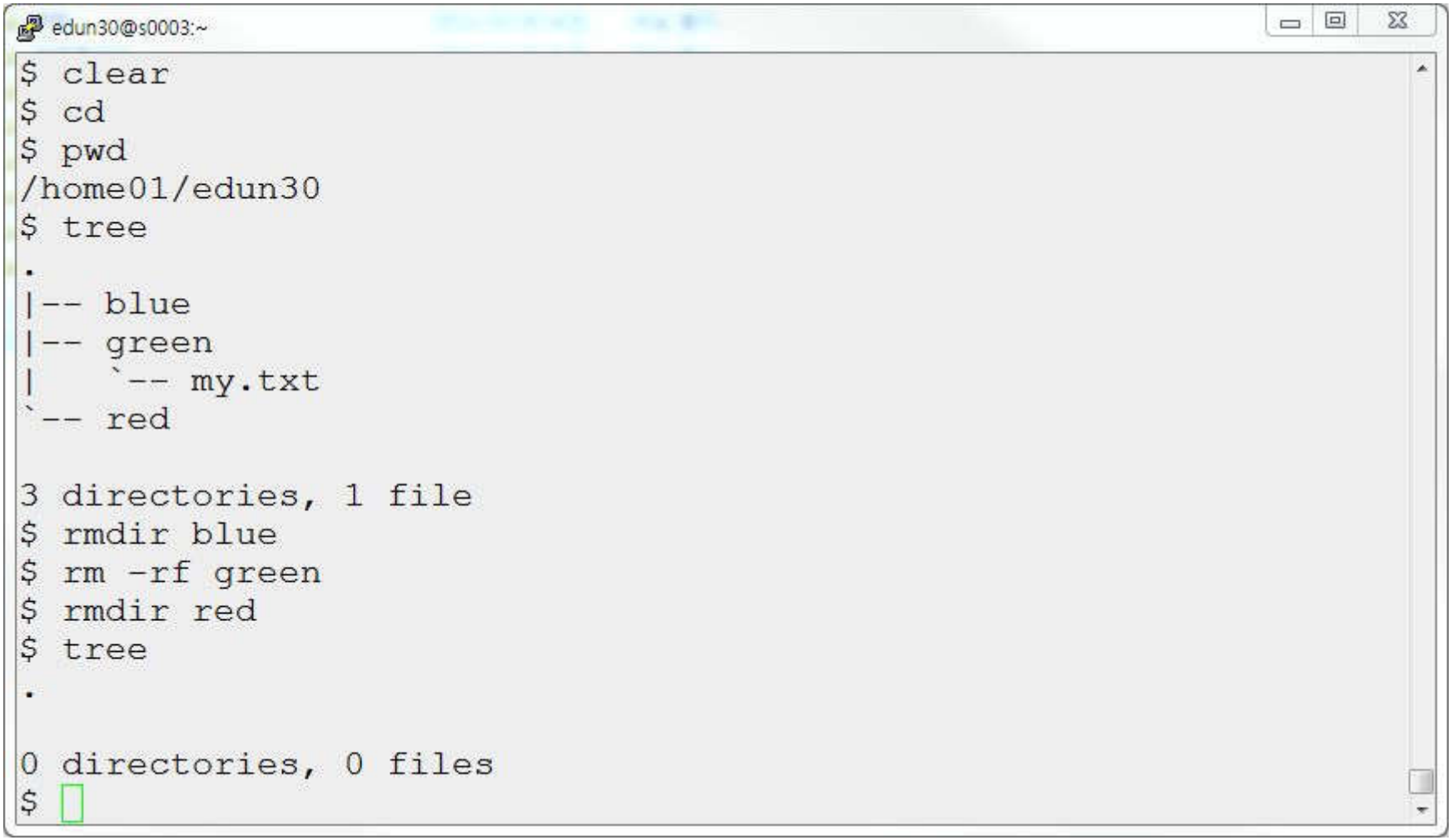
➢기본 사용법 **(rm, rmdir)**

```
edun30@s0003:~/red
$ clear
$ cd
$ pwd
/home01/edun30
$ cd red
$ pwd
/home01/edun30/red
$ ls
white/
$ rmdir white
rmdir: white: Directory not empty
$ cd white
$ ls
my.txt
$ rm my.txt
$ ls
$ cd ..
$ rmdir white
$ ls
$
```

디렉터리안에 파일이 있으면 삭제불가

디렉터리 이동/생성/삭제 관련

➢기본 사용법 **(rm –rf : 파일이 포함된 디렉터리 지우기)**



디렉터리 / 파일 관련

➢ **rm**
- 파일이나 디렉터리 (-rf 옵션)를 삭제
- 옵션
  - -i : 삭제시 확인
  - -f : 강제 삭제
  - -r : 디렉터리를 삭제할 때 하위 디렉터리와 파일도 모두 삭제
  - -v : rm 명령어 진행 과정 출력

➢ **mv**
- 파일과 디렉터리의 이름을 변경하거나 경로를 옮길 때 사용
- -b 옵션 사용시 : 같은 파일이 있을 경우는 backup 파일을 생성

디렉터리 / 파일 관련

➢기본 사용법 **(rm, mv)**

```
edun30@s0003:~
$ cd; pwd
/home01/edun30
$ mkdir cyan
$ touch my.txt
$ mv my.txt ./cyan
$ cd cyan
$ ls -l
total 0
-rw-r----- 1 edun30 edungrp 0 Nov  1 15:04 my.txt
$ cp -p my.txt my2.txt
$ ls -l
total 0
-rw-r----- 1 edun30 edungrp 0 Nov  1 15:04 my2.txt
-rw-r----- 1 edun30 edungrp 0 Nov  1 15:04 my.txt
$ mv my.txt ../you.txt
$ cd ..
$ ls -l
total 4
drwxr-x--- 2 edun30 edungrp 4096 Nov  1 15:04 cyan/
-rw-r----- 1 edun30 edungrp    0 Nov  1 15:04 you.txt
```

디렉터리 / 파일 관련

➢기본 사용법 **(rm, mv)**

edun30@s0003:~

```
$ clear
$ cd; pwd
/home01/edun30
$ ls -l
total 4
drwxr-x--- 2 edun30 edungrp 4096 Nov  1 15:04 cyan/
-rw-r----- 1 edun30 edungrp    0 Nov  1 15:04 you.txt
$ rm -i you.txt
rm: remove regular empty file `you.txt'? y
$ ls -l cyan
total 0
-rw-r----- 1 edun30 edungrp 0 Nov  1 15:04 my2.txt
$ rm -rf cyan
$ ls -l
total 0
$
```

디렉터리 / 파일 관련

➢ **cat**

- 간단한 텍스트 파일을 생성 하거나 텍스트 파일 내용을 확인

ex) cat /etc/passwd

➢ **echo**

- 텍스트를 화면 상에 출력

ex) echo “hello”

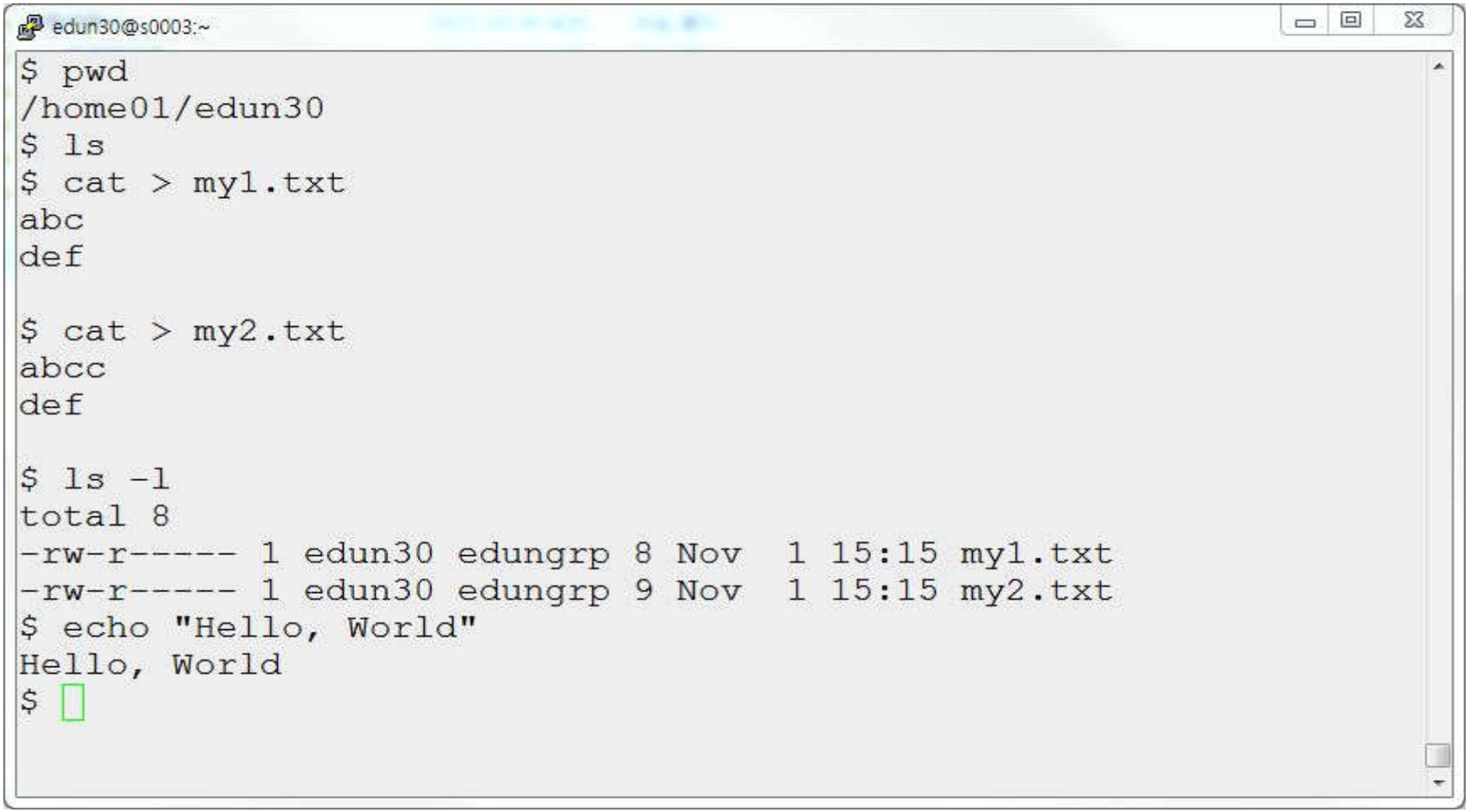
echo $HOME

```
2927% [edun60@ibs0001 ~]# echo "hello"
hello
2928% [edun60@ibs0001 ~]# echo $HOME
/home01/edun60
2929% [edun60@ibs0001 ~]# echo $PWD
/home01/edun60
```

디렉터리 / 파일 관련

➢기본 사용법 **(cat, echo)**

- cat 명령어로 text 파일 작성 후 종료는 ***Ctrl + c*** 를 누른다



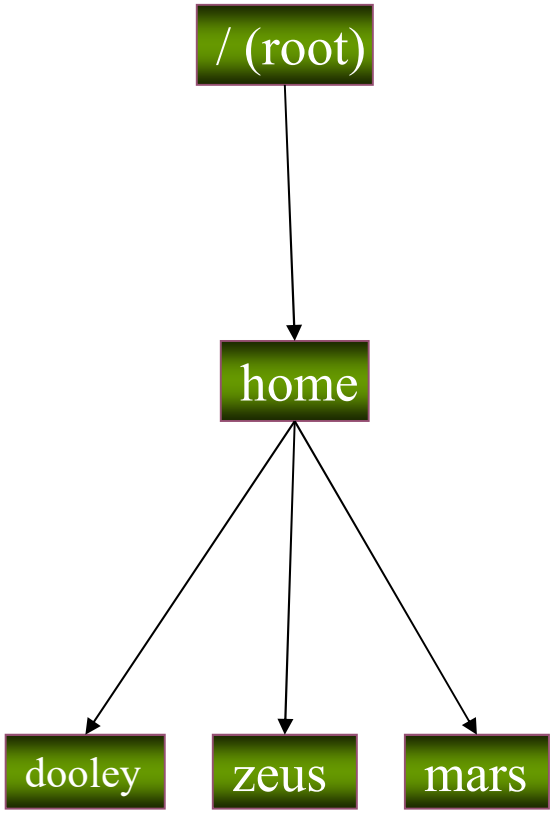
Absolute and Relative Path

- 절대 경로**(Absolute Path)**
  - /(root directory) 기준

- 상대 경로**(Relative Path)**
  - .(current directory) 기준

- **(e.g) zeus -> mars**
  - cd /home/mars
  - cd ../mars

- **Home Directory**
  - cd ~



which, whereis, whatis

➢**which**

- 명령어의 경로를 보여준다

```
$ which  ls
alias ls='ls -F --show-control-chars --color=auto'
      /bin/ls
$ which  cal
/usr/bin/cal
```

➢**whereis**

- which와 비슷한 명령어로, 실행파일, 소스, man page의 경로를 보여준다

```
$ whereis  ls
ls: /bin/ls /usr/share/man/man1p/ls.1p.gz /usr/share/man/man1/ls.1.gz
$ whereis  cal
cal: /usr/bin/cal /usr/share/man/man1p/cal.1p.gz /usr/share/man/man1/cal.1.gz
```

➢**whatis :** 명령어가 무슨 일을 하는지 설명 해준다

```
$ whatis  cal
cal             (1)  - displays a calendar
cal             (1p)  - print a calendar
```

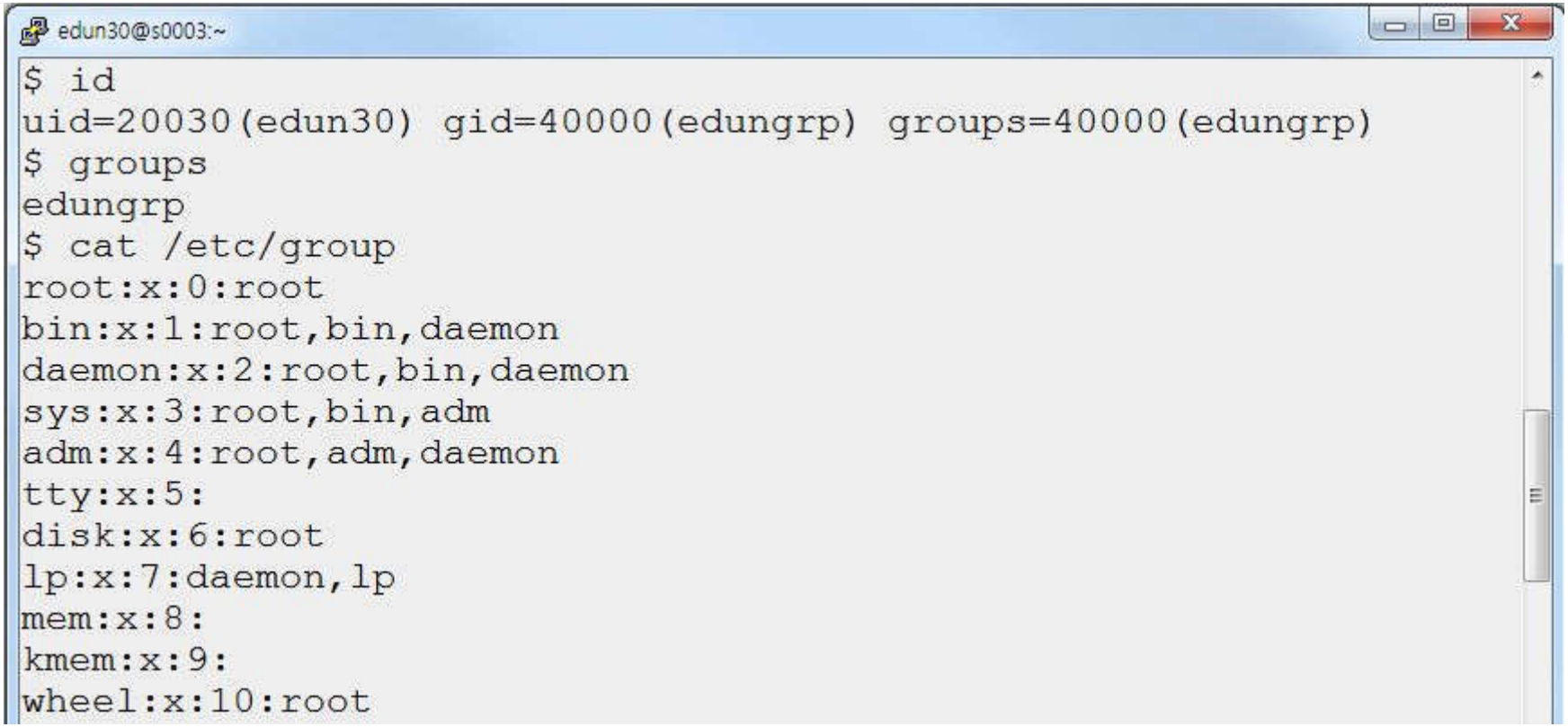
id, groups

➢**id**

- 자신에 대한 uid, gid에 대한 정보

➢**groups**

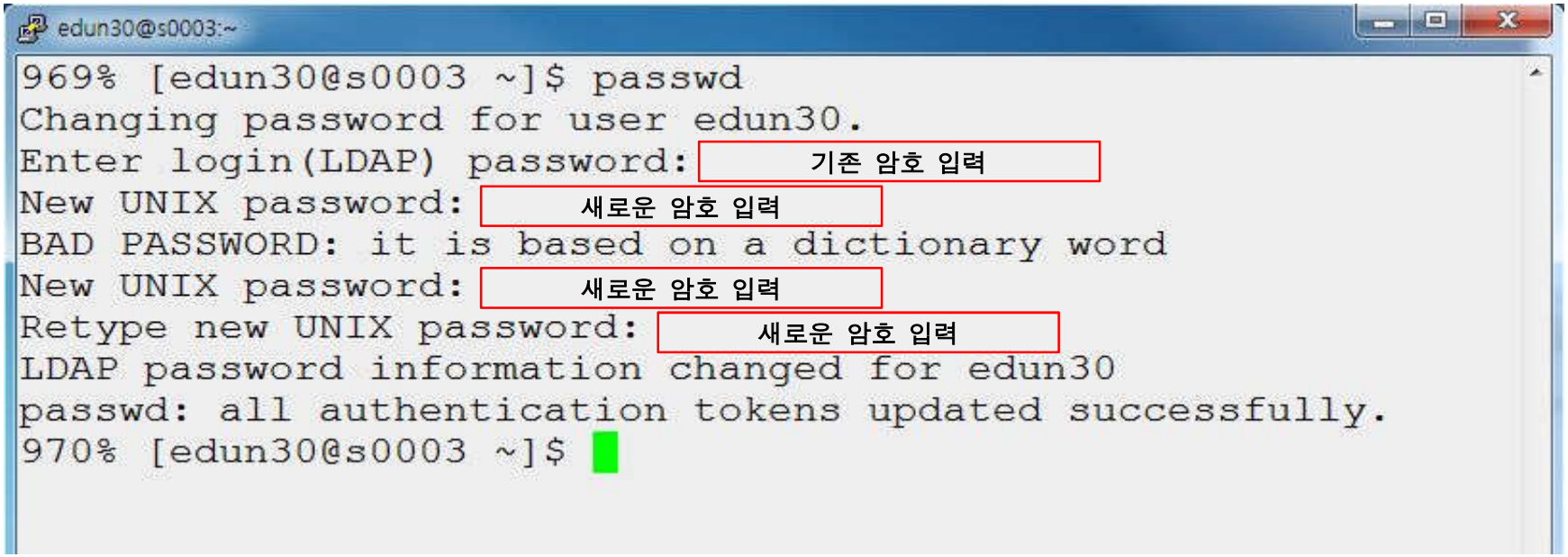
- /etc/group 파일을 참고해서 자신이 속한 그룹을 보여준다



passwd

➢사용자의 암호를 변경한다

- 주기적으로 변경해 주는 것이 좋다
- 해킹되기 쉬운 너무 쉬운 암호는 피하고, 특수문자를 섞어서 쓰는 것이 일반적이다
- 아래 예는 사전적으로 쓰이는 단어를 사용해서 메시지를 뿌려준 화면이다

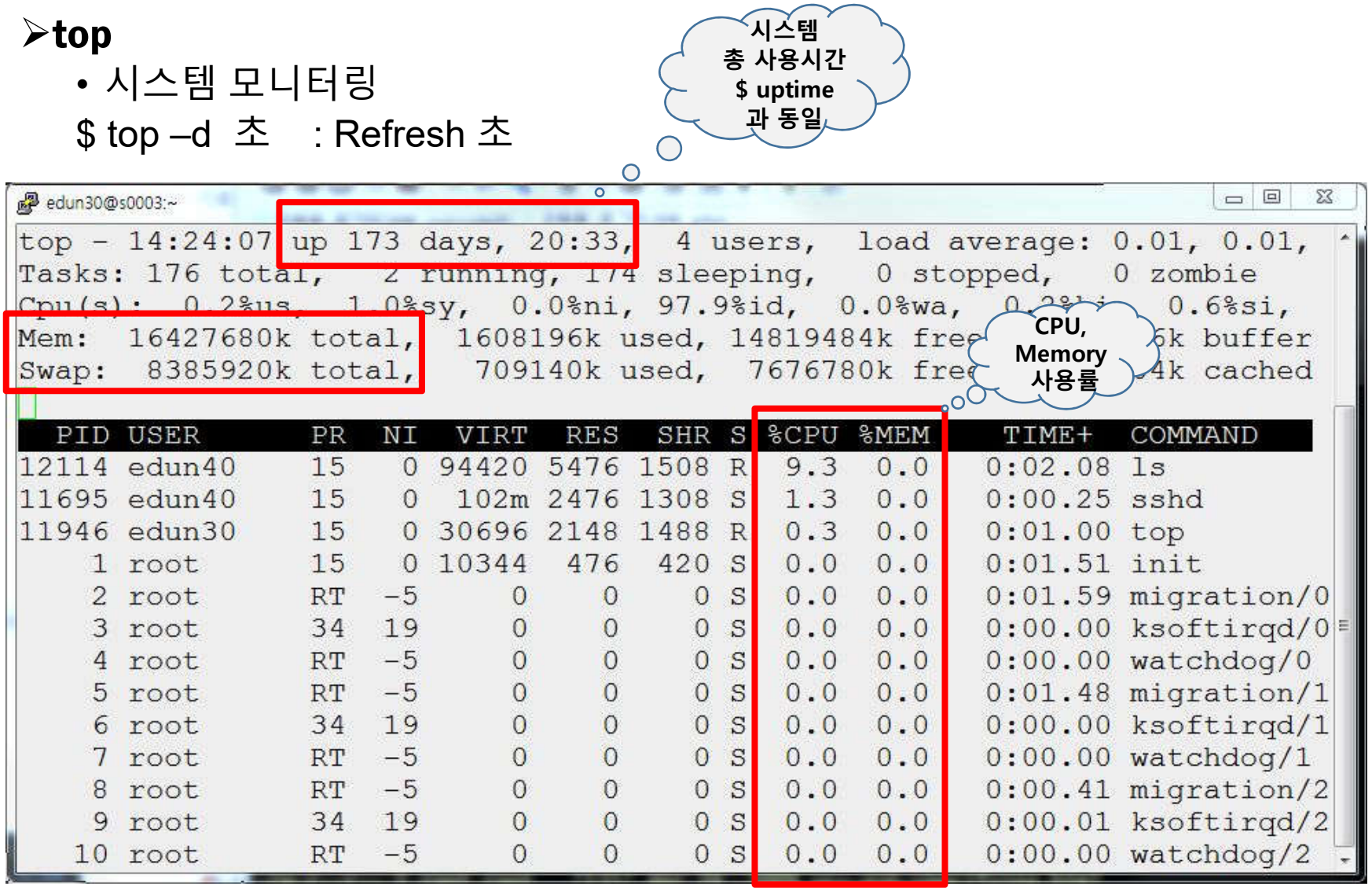


top

➢**top**

- 시스템 모니터링

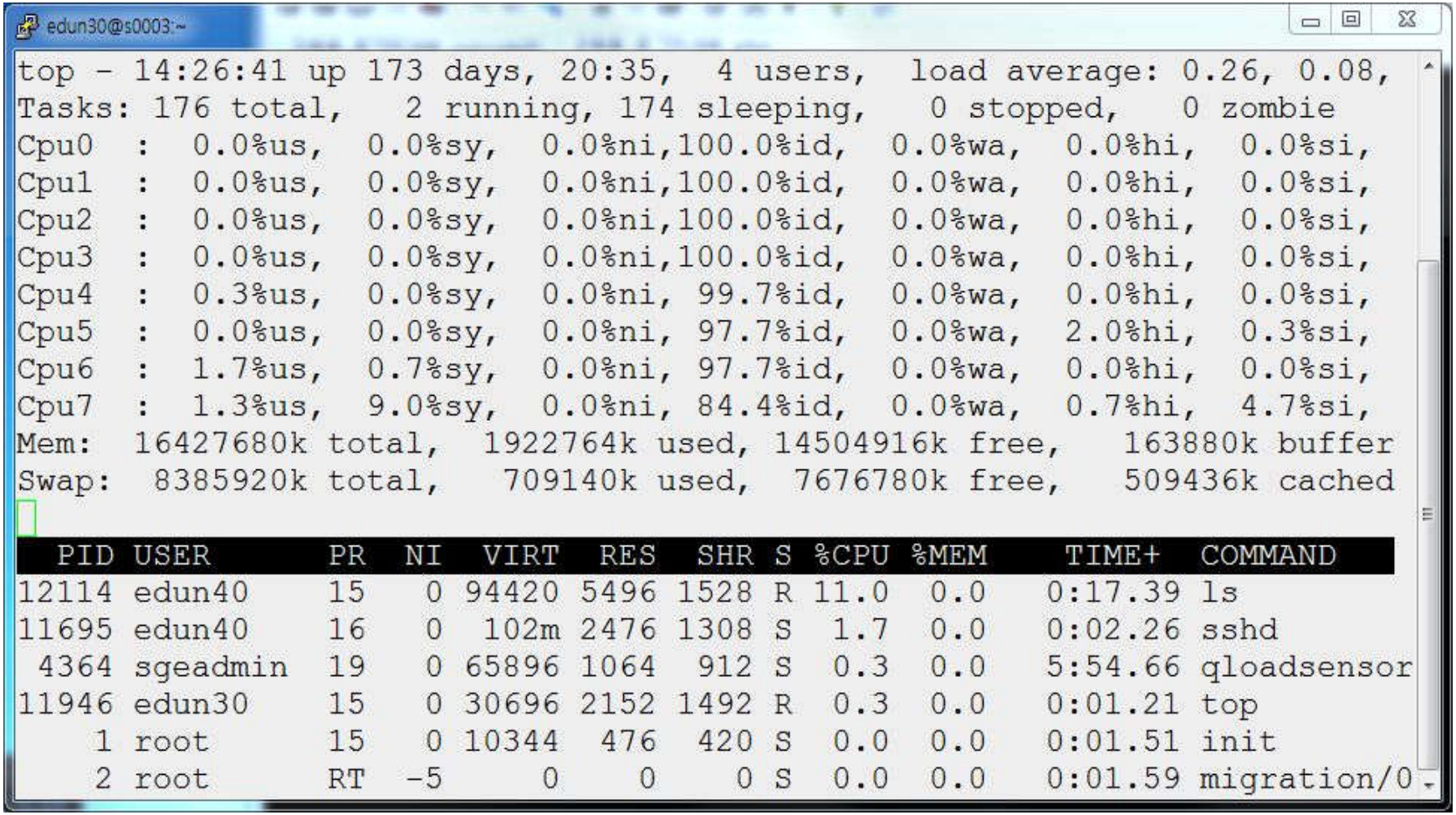
$ top –d 초 : Refresh 초



top

➢top

- CPU 상태 보기 (press ‘1’ and ‘q’ to exit)



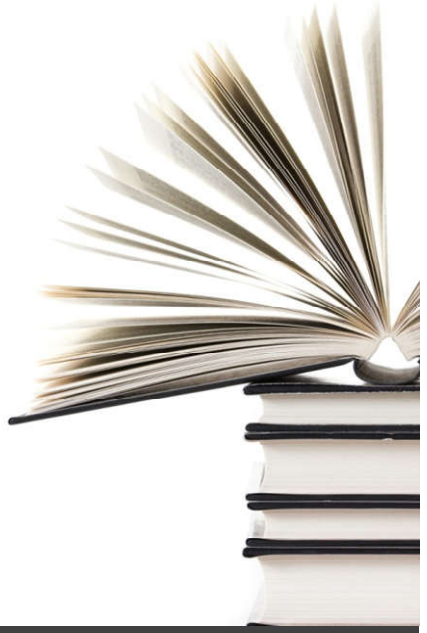
history

➢history

- 리눅스는 사용자가 사용했던 명령어들을 기억하고 있으며, 필요할 땐 언제나 다시 불러서 사용할 수 있다. 사용자 홈 디렉터리의 .bash_history 파일에 저장돼 있다
- 방향 Key (Up, Down) 사용

```
$ history
 1003  12-11-01 14:11:14 who
 1004  12-11-01 14:11:15 id
 1005  12-11-01 14:11:15 groups
 1006  12-11-01 14:11:18 history
$ !1004                                          // 1004번째 수행한 명령어를 수행
id
uid=20030(edun30) gid=40000(edungrp) groups=40000(edungrp)
$ !!                                             // 가장 최근에 사용한 명령어를 수행
$ history 5                                      // 가장 최근에 사용한 명령어 5개를 출력
$ !wh                                            // wh 라는 문장이 포함된 가장 최근에 사용한 명령어를 수행
                                                    위의 history에서는 wh가 포함된 who를 수행하게 된다
```

VI Editor

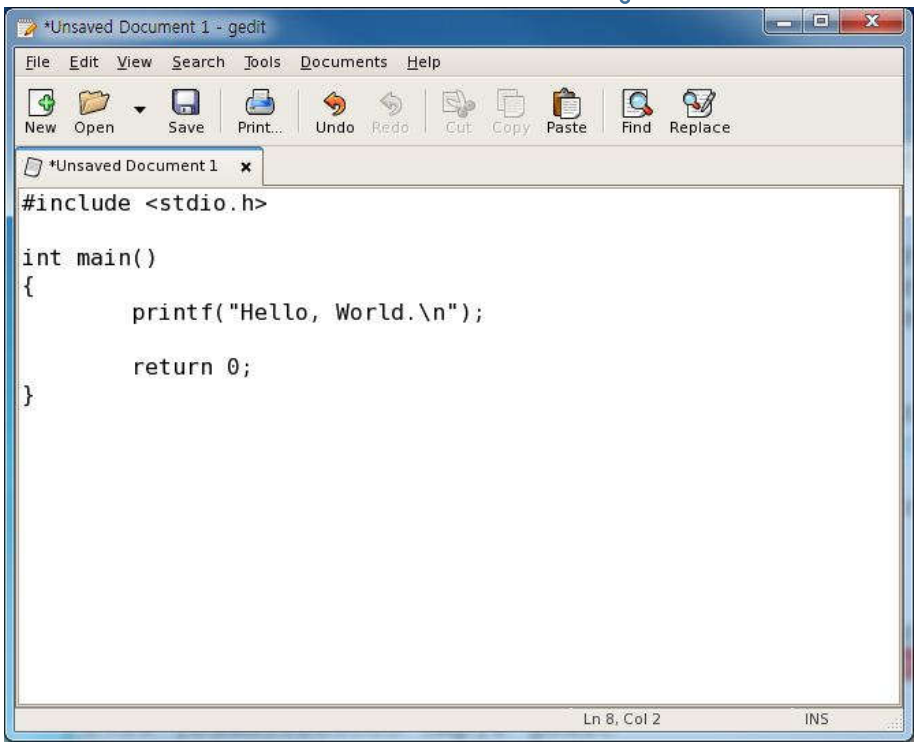


편집기 (Editor)

➢**vim(vi)**
- 가장 기본적인 텍스트 에디터, OS에 기본적으로 포함

➢**emacs**
- 강력하고 기능이 다양한 에디터, 사용법이 어렵고 복잡

➢**gedit**
- GUI 기반의 텍스트 에디터
- 메모장과 비슷해서 사용이 쉬움



vi 에디터란?

➢**vi editor**

- VIsual display editor 를 의미
- 처음 접하는 사람에게 쉽지 않음
- 대부분의 유닉스 계열 시스템에 설치 되어 있음
- 워드 프로세서의 기능의 상당 부분을 가지고 있음
- vim 이라는 vi 의 클론이 포함되어 있음

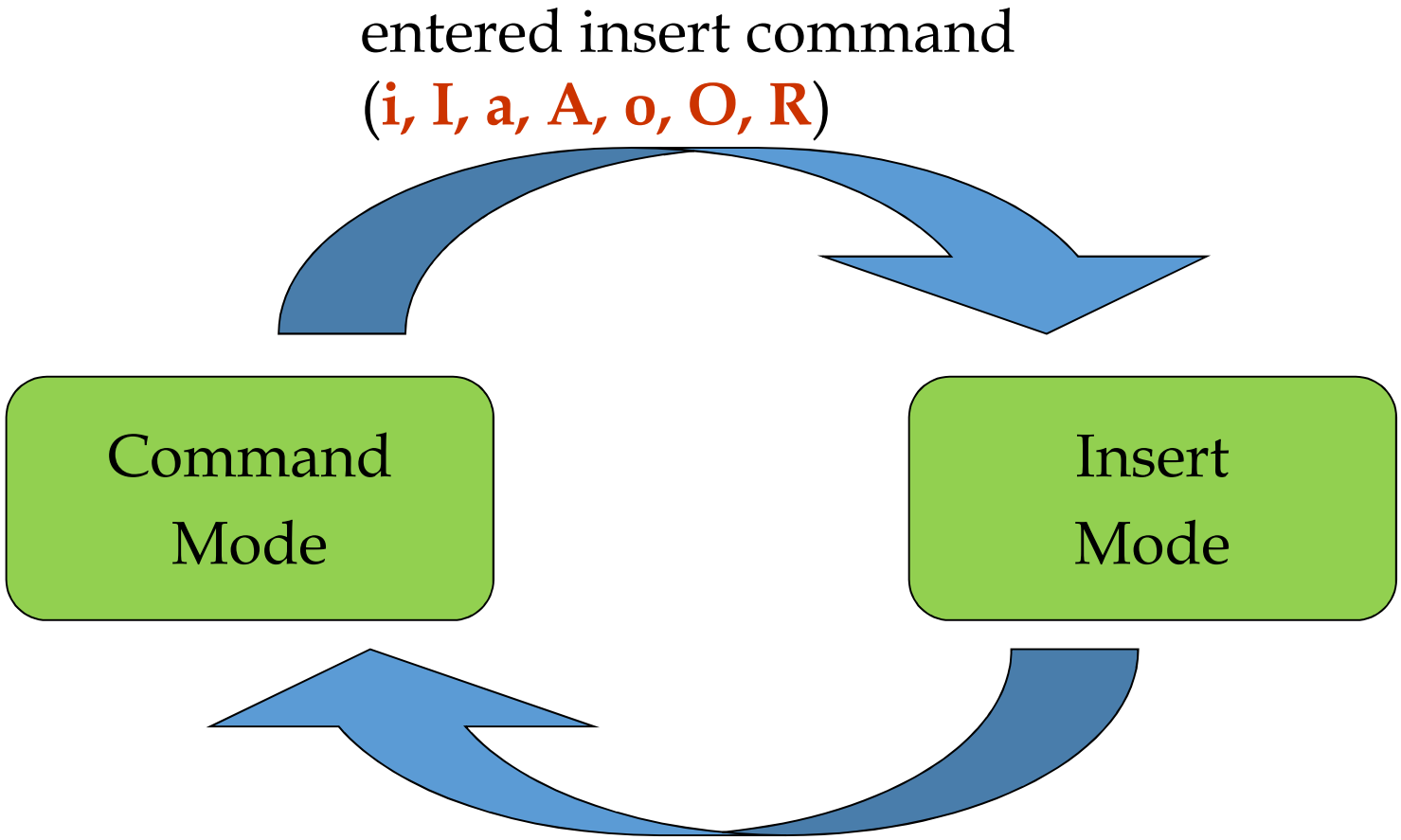
파일 열기

➢Open a file

- $ vi file (편집 모드)
- $ view file (읽기 모드)

➢modes

- 입력 모드
  - 입력하는 모든 것이 편집 버퍼에 입력됨
  - 입력모드에서 빠져 나올 때(명령행 모드로 변경 시) : “ESC” key
- 명령행 모드
  - 입력하는 모든 것이 명령어 해석됨


entered insert command
(i, I, a, A, o, O, R)
Command Mode
Insert Mode
entered <ESC>

이동 및 화면 제어

➢기본 단위 커서 이동

h	**커서를 한 문자 왼쪽으로 이동**
j	**커서를 아래 줄 같은 컬럼으로 이동**
k	**커서를 위 줄 같은 컬럼으로 이동**
l	**커서를 한 문자 오른쪽으로 이동**
<ENTER>	**커서를 다음 줄의 첫 문자로 이동 (10ENTER)**
방향키	

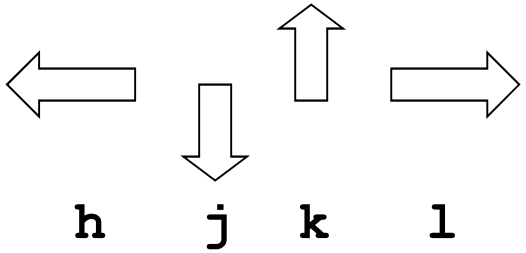
➢화면 단위 커서 이동

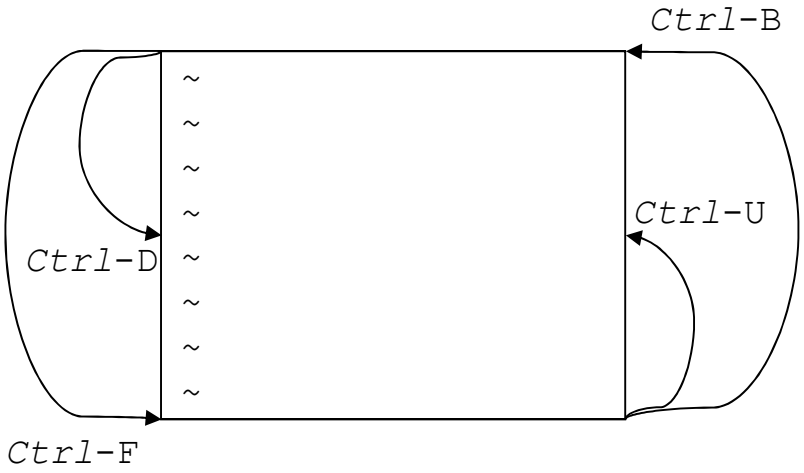
H	**화면의 맨 위쪽으로 커서 이동**
M	**화면의 중간 줄로 커서 이동**
L	**화면의 맨 아래 줄로 커서 이동**

화면 커서 이동

➢Cursor Movement

- nn 번 줄로 이동 : nnG
- nn 줄 아래로 이동: nn<Enter>





이동 및 화면 제어

➢단어, 문장, 절 단위 이동

w	다음 단어의 첫 문자로 커서 이동
b	커서가 위치한 앞 방향의 단어로 커서 이동
e	커서가 위치한 단어의 끝으로 커서 이동
)	다음 문장으로 커서 이동
}	다음 절로 커서 이동
(	)와 반대
{	}와 반대
0	현재 줄의 첫 문자로 커서 이동
^	현재 줄의 처음으로 커서 이동 (탭이나 공백이 아닌)
$	현재 줄의 마지막으로 커서 이동

이동 및 화면 제어

➢단어, 문장, 절 단위 이동

^F	한 화면 아래로 이동
^B	한 화면 위로 이동
^D	반 화면 아래로 이동
^U	반 화면 위로 이동
:$	마지막 줄로 이동
G	nG : 줄 번호 n으로 이동 1G : 편집 버퍼의 첫 줄로 이동하기 G : 편집 버퍼의 마지막 줄로 이동하기

편집 모드

➢데이터 첨가, 삽입, 치환, 해제

a	현재 커서 위치 오른쪽부터 데이터 첨가
i	현재 커서 위치의 왼쪽부터 데이터 첨가
o	현재 줄 아래 줄을 공백 줄로 만듦
A	현재 줄의 마지막에 데이터 첨가
I	현재 줄의 첫 문자 앞에 문자 삽입
O	현재 줄의 윗 줄을 공백 줄로 만듦
r	현재 커서가 위치한 한 문자 치환
R	현재 커서 위치에서 <esc>키를 칠 때까지 문자 치환
cw	현재 커서 위치의 단어를 다른 단어로 치환
C	현재 커서의 위치부터 그 줄의 마지막까지 치환
s	현재 커서의 문자를 치환
S	현재 커서의 줄을 치환
cc	현재 커서가 위치한 줄을 다른 내용으로 치환
~	현재 커서 위치의 한 문자를 소문자, 대문자로 전환
<ESC>	데이터 입력 모드에서 벗어남

편집 모드

➢데이터 삭제

x	현재 커서가 위치한 한 문자삭제
X	현재 커서가 위치한 앞 문자를 삭제
dw	현재 커서가 위치한 오른쪽 단어 삭제
db	현재 커서가 위치한 왼쪽 단어 삭제
dd	현재 커서가 위치한 줄 삭제
D	현재 커서가 위치한 곳에서 오른쪽의 내용삭제
:n1, n2 d	n1 ~ n2 라인을 삭제

➢명령의 취소 및 반복

u	바로 앞의 행한 명령 취소
U	한 줄 내에서 행한 명령 취소
.	바로 앞에 행한 명령 재수행

편집 모드

➢데이터 이동과 복사

yw	현재 커서가 위치한 단어를 버퍼에 복사
yy 10yy	현재 줄이 버퍼에 복사 10줄을 복사
nY	현재 커서 위치로부터 n 줄 만큼 복사
p	현재 커서 오른쪽 또는 아래 줄에 버퍼 내용 복사
P	현재 커서의 왼쪽 또는 위 줄에 버퍼 내용 복사
: n1, n2 y	n1부터 n2 라인까지 복사

➢줄의 결합

J	현재 줄과 다음 줄을 연결

명령 실행 모드

➢vi 종료

:x ZZ	파일을 디스크에 저장한 후 vi를 벗어난다
:wq	파일을 디스크에 저장한 후 vi를 벗어난다
:wq 파일명	기존 파일명을 새로운 파일명에 저장하고 vi 벗어남
:q!	파일을 디스크에 저장하지 않고 vi에서 벗어남

➢기타 명령어

:set nu	vi 상태에 있는 파일에 줄 번호를 부여
:set nonu	파일에 있는 줄 번호 취소

Example 1

➢예제 문장

I hope that the KISTI Supercomputing Center is known as the place where the dreams of Korean scientists and engineers are becoming a reality.

I hope that the Center contributes to furthering national public welfare and also to opening up a bright future for the wold.

It is my pleasure to watch as these dreams are realized through scientists' creativity and engineers' efforts using the tool of supercomputing.

Example 1

➢예제 문장

1. 앞 예제 에서 작성한 파일 이름을 contact1.txt 변경
2. contact1.txt 를 contact2.txt 로 복사
3. contact1.txt 와 contact2.txt 를 novel.txt 파일로 합치기
4. wold -> world 로 수정 (검색 후)
5. 4, 5번째 줄 삭제

VI Editor Practise

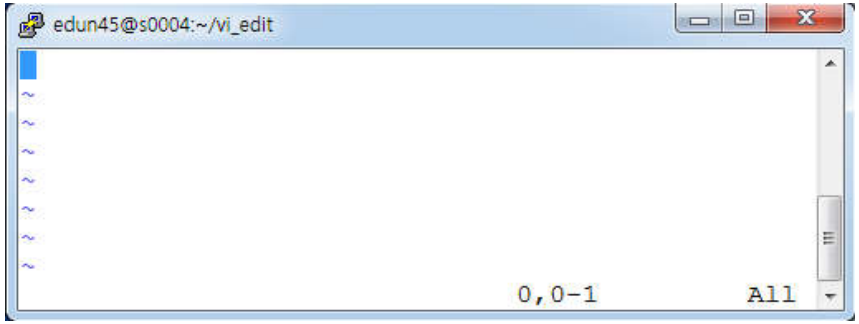
➢ 1. 파일 생성 및 편집

- 처음에는 명령 모드로 실행
- 편집 모드로 전환해야 글자 편집 가능
- ‘a’, ‘I’를 누르면 명령모드에서 편집모드로 전환됨
  - ‘a’: 현재 커서 위치에서 오른쪽부터 데이터 첨가 (**a**ppend)
  - ‘i’ : 현재 커서 위치에서 왼쪽부터 데이터 첨가 (**i**nsert)

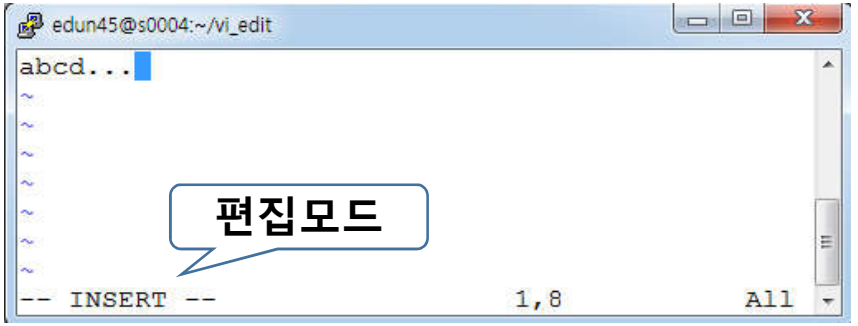
vi 편집기로 파일 생성

```
$ vi filename
```

명령 모드



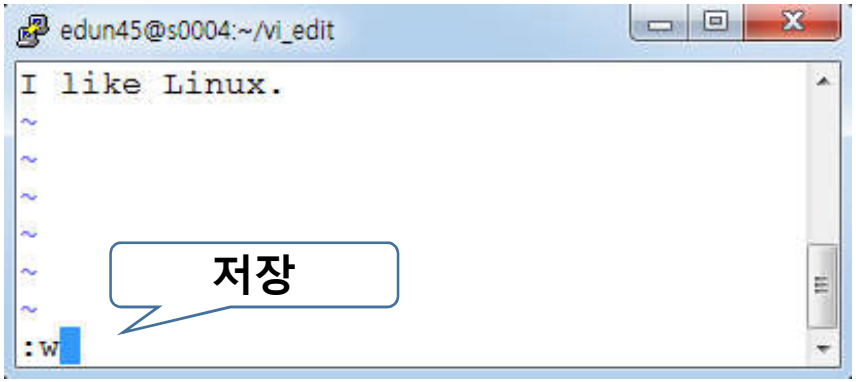
편집 모드

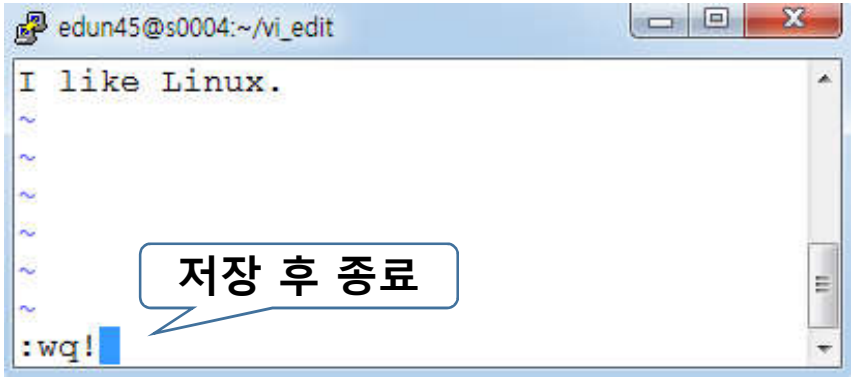


VI Editor Practise

- **2.** 저장 및 종료
  - 편집 모드에서 명령 모드로 전환 (‘ ESC ‘ 키 누름)
  - 저장 : **‘ ESC ’ + ‘ :w ’**
  - 저장하지 않고 종료 : **‘ ESC ’ + ‘ :q ’**
  - 저장 후 종료 : **‘ ESC ’ + ‘ :wq! ’**

(‘ wq ’ 명령은 저장 후 종료, ‘ ! ‘ 는 강제 저장, 종료를 의미)





➢ 3. 편집

– 명령 모드에서 실행

1) 삭제

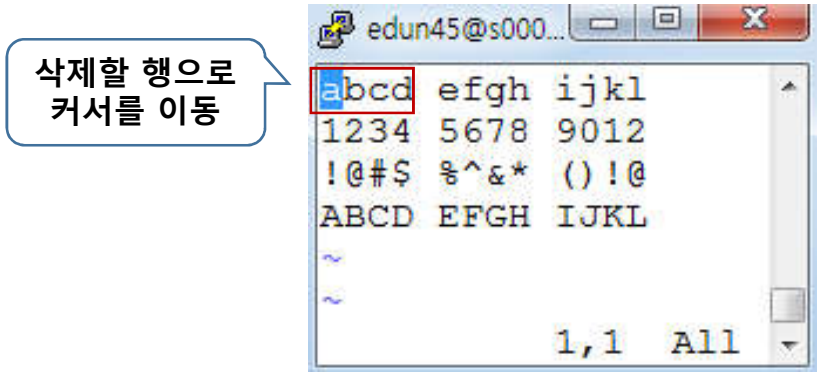
x : 커서 위치의 글자 삭제

dw : 한 단어를 삭제

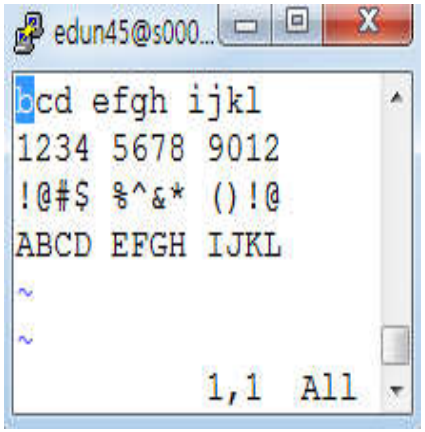
dd : 커서가 있는 행을 삭제

3dd : 커서로부터 3개의 행이 삭제

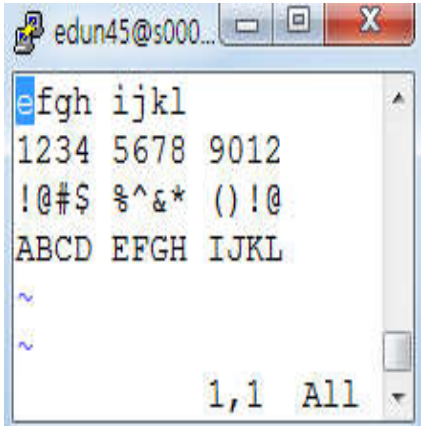
삭제 실행 전



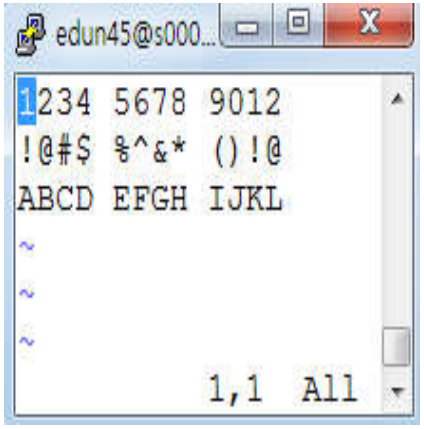
한 글자 삭제 (x)



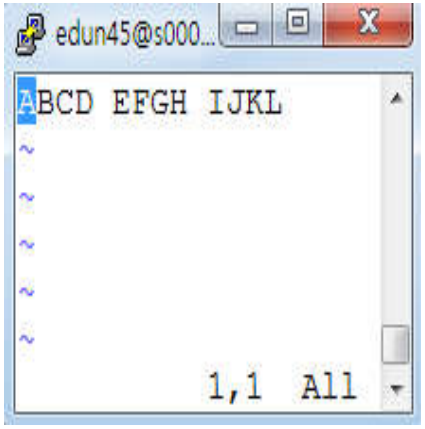
한 단어 삭제 (dw)



한 행 삭제 (dd)



3행 삭제 (3dd)



➢ 3. 편집

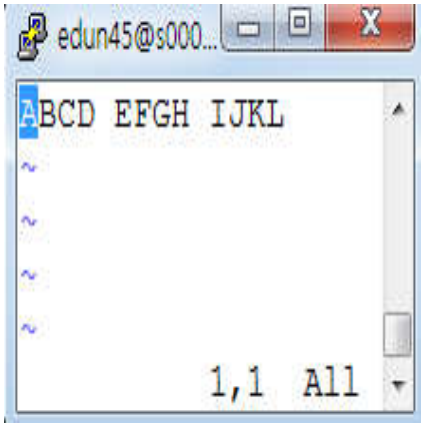
– 명령 모드에서 실행

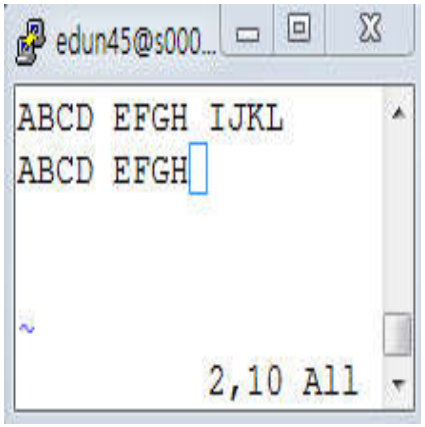
2) 복사와 붙이기

yw : 현재 커서 위치의 단 단어를 복사
3yw : 현재 커서 위치에서 3개의 단어를 복사
yy : 커서 위치의 한 행을 복사
3yy : 커서 위치에서 3 행을 복사

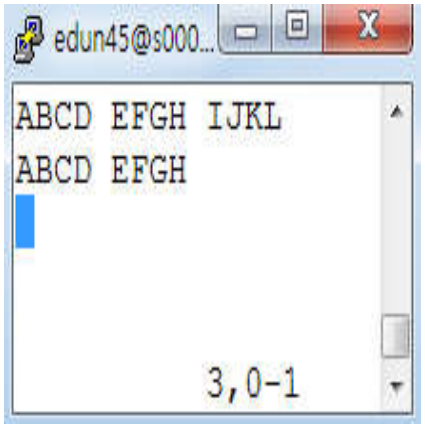
p : 복사한 내용을 붙여 넣음
2p : 복사한 내용을 2번 붙여 넣음

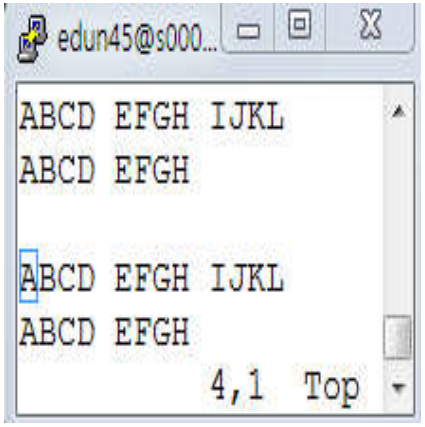
1) 2단어 복사(2yw) 커서 이동 붙여 넣기 (p)





2) 2 행 복사(2yy) 커서 이동 붙여 넣기 (p)

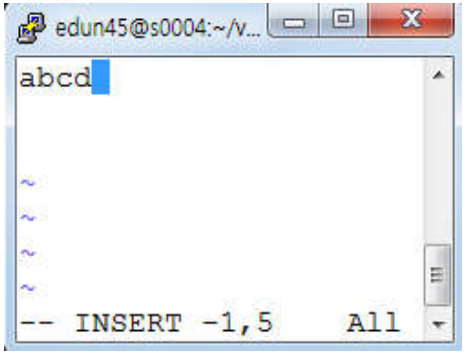




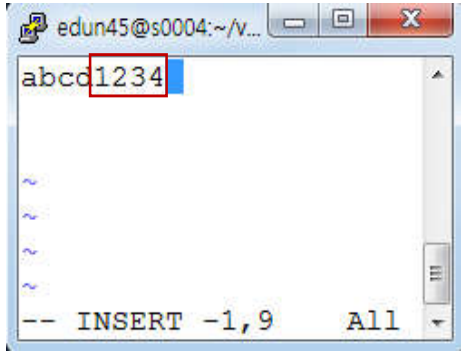
VI Editor Practise

- **4.** 되돌리기와 되살리기
  - 명령 모드에서 실행
  - 되돌리기 : **‘ u ’**
  - 되살리기 : **‘ Ctrl ’ + ‘ r ’**

1) 수정 전

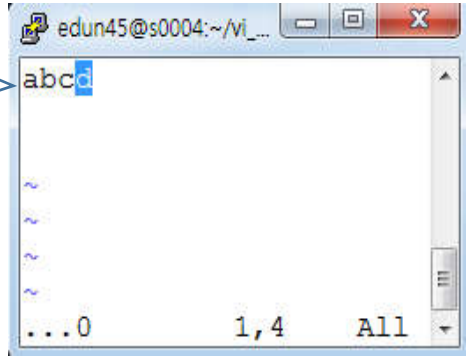


2) 수정 후



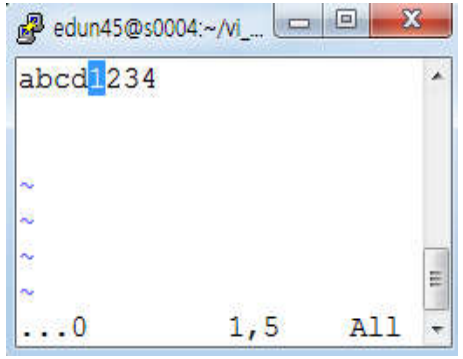
명령모드 전환(ETC)

3) 되돌리기 수행



수정하기 전 상태로 돌아감

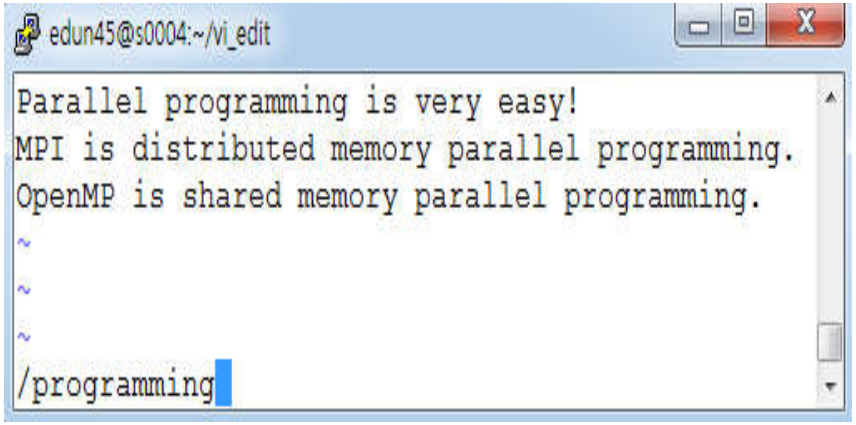
4) 되살리기 수행



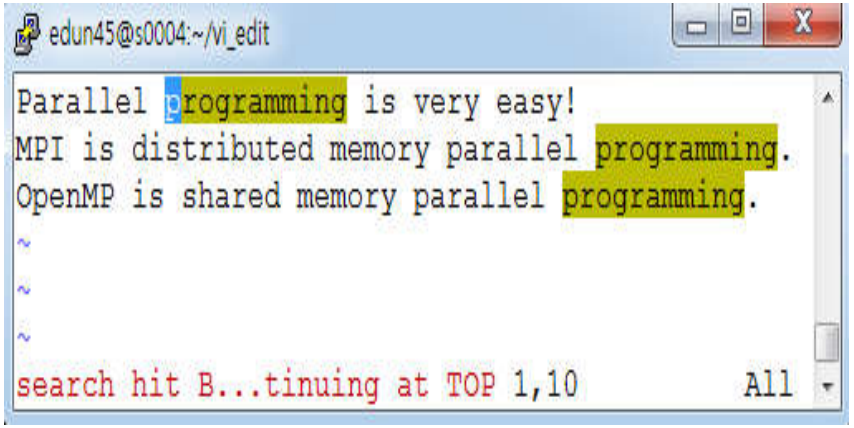
VI Editor Practise

- **5.** 문자열 탐색
  - 명령 모드에서 실행
  - **/** [찾고자 하는 문자열] 또는 **?** [찾고자 하는 문자열]
  - 실행 후 ‘ **n** ’ 을 누르면 다음 매칭 문자열로 이동
  - 실행 후 ‘ **N** ’ 을 누르면 이전 매칭 문자열로 이동

문자열 탐색 실행



검색된 문자열 표시

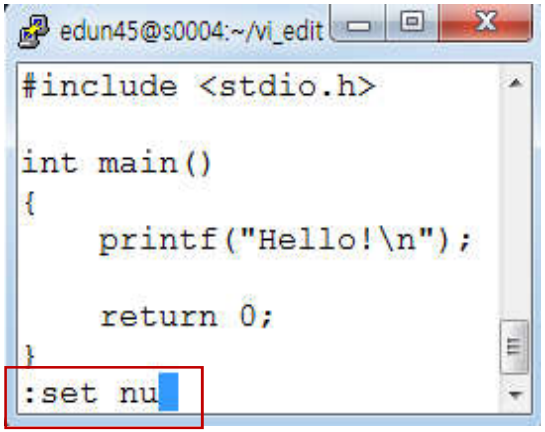


VI Editor Practise

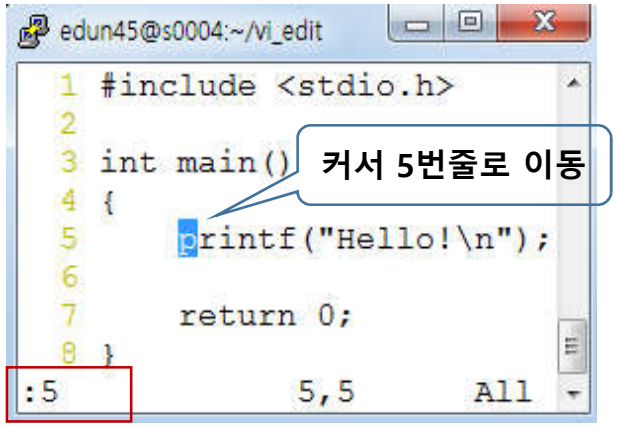
- **6.** 줄 번호 표시 및 이동
  - 명령 모드에서 실행
  - 줄 번호 표시 : ‘ **:set nu** ’
  - 줄 번호 삭제 : ‘ **set nonu** ’
  - 줄 번호 이동 : ‘ **: 줄번호** ’

(‘ wq ’ 명령은 저장 후 종료, ‘ ! ‘ 는 강제 저장, 종료를 의미)

줄 번호 표시



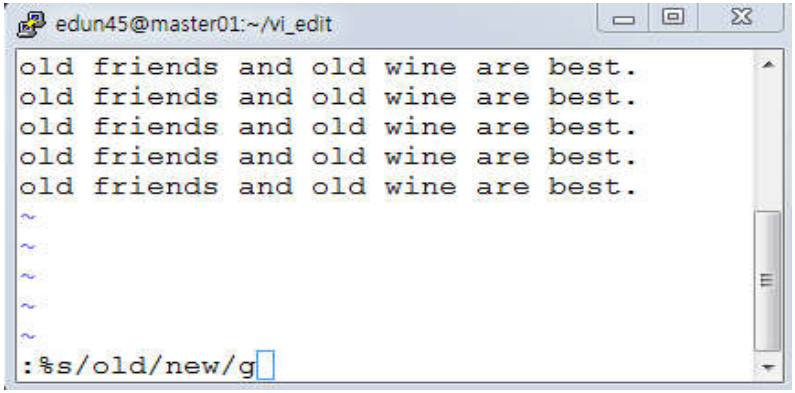
줄 번호 이동



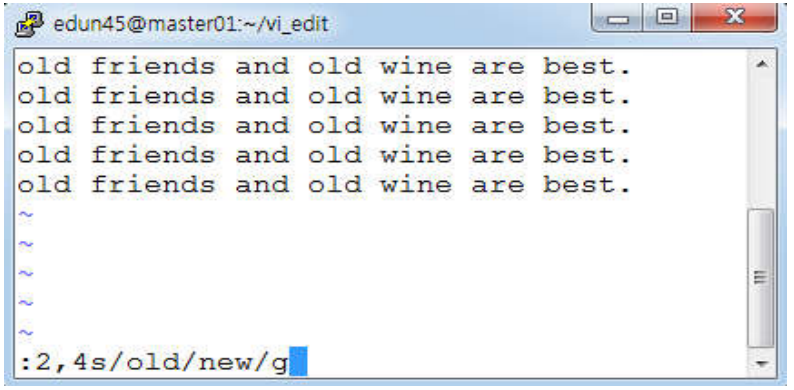
VI Editor Practise

- **7.** 문자열 치환
  - 명령 모드에서 실행
  - 치환 명령어 : **‘ %s/old/new/g ’**

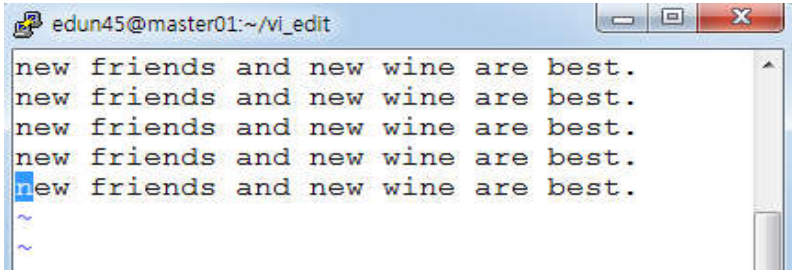
전체 범위의 ‘old’문자를 ‘new’로 치환



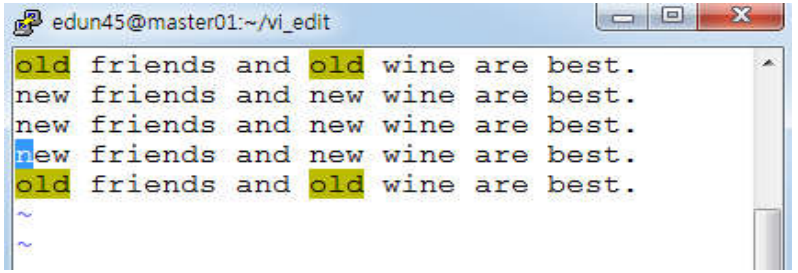
2행부터 4행까지 문자열 치환



전체 범위 치환 결과

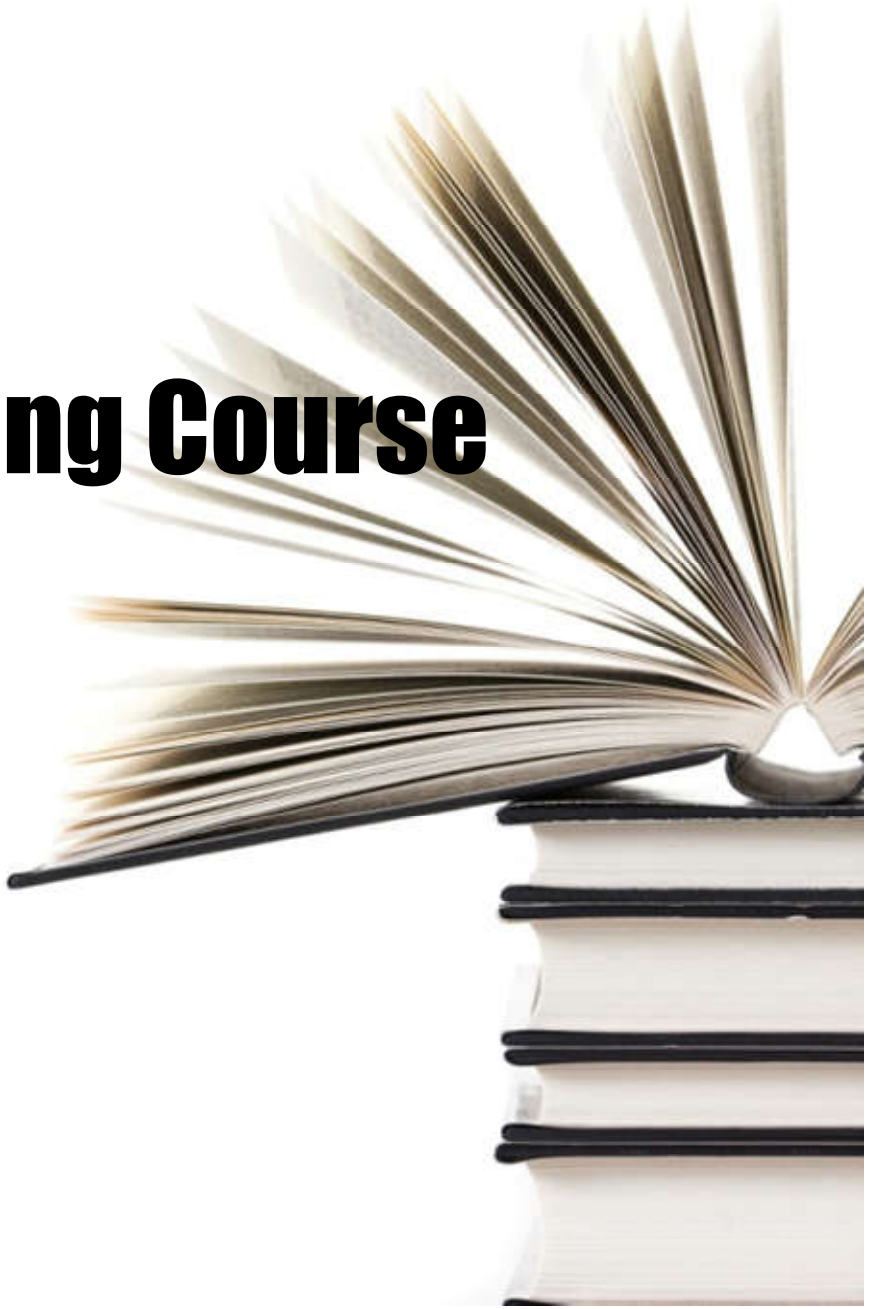


2행부터 4행까지 치환 결과



KiSTi SUPERCOMPUTING EDUCATION CENTER
www.kisti.re.kr

OpenMP Training Course
in PNU



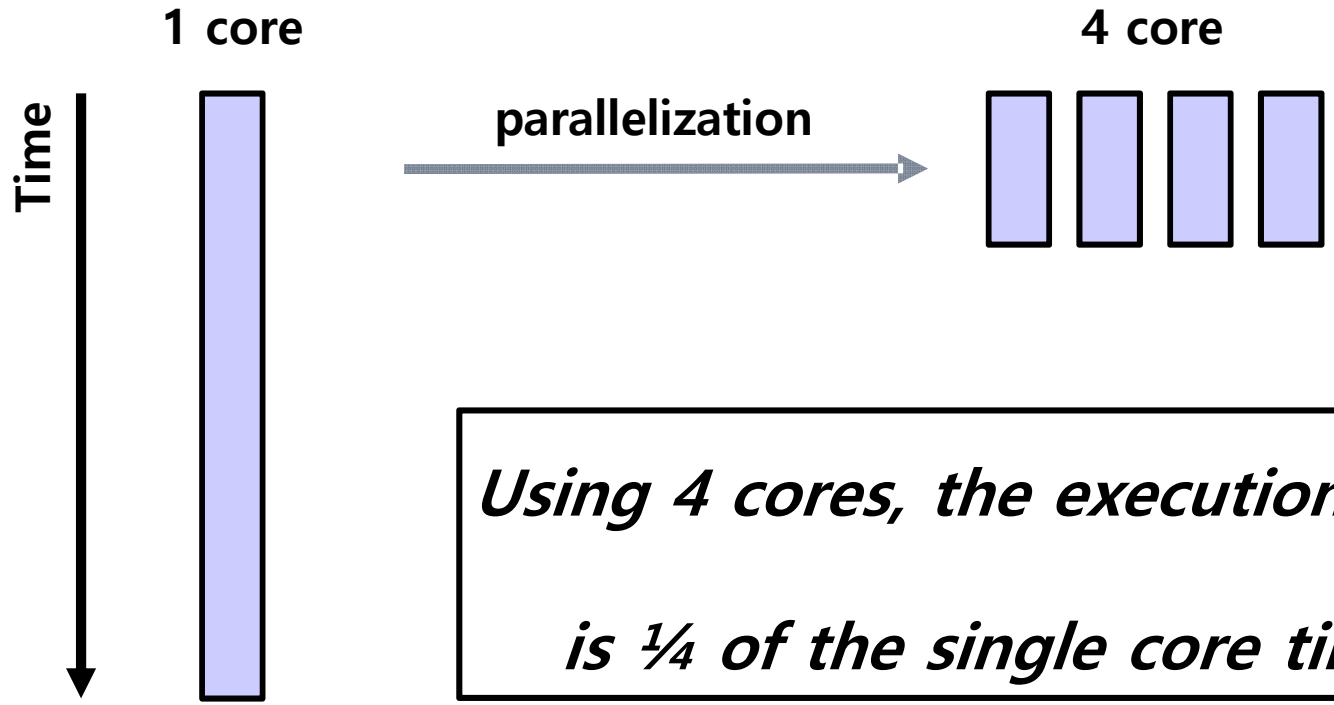
http://www.ksc.re.kr
http://edu.ksc.re.kr

National Institute of Supercomputing and Networking 1967

Why Parallelization?

Parallelization is another optimization technique
The goal is to reduce the execution time

To this end, multiple processors, or cores, are used

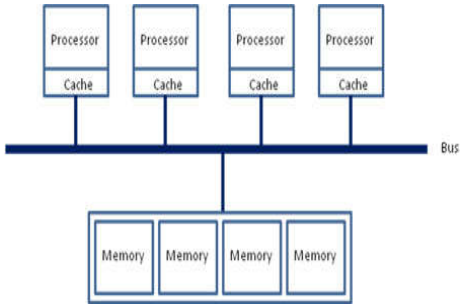


Using 4 cores, the execution time is ¼ of the single core time

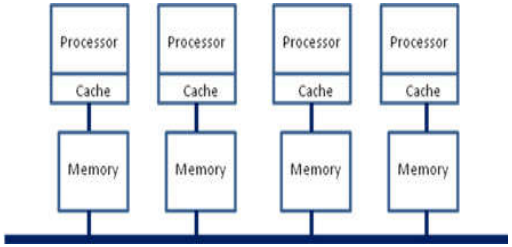
Parallel Computing

➢Shared Memory

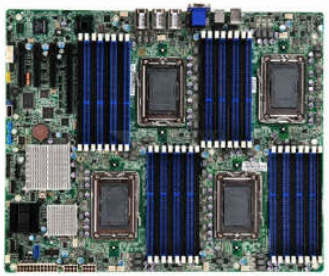
- Single address space for all processors



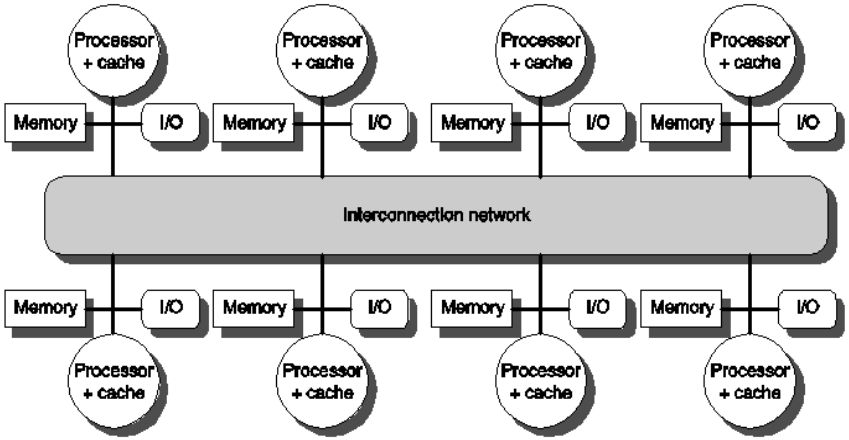
<UMA>

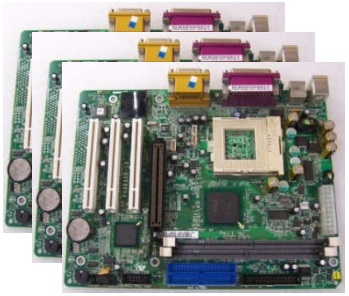


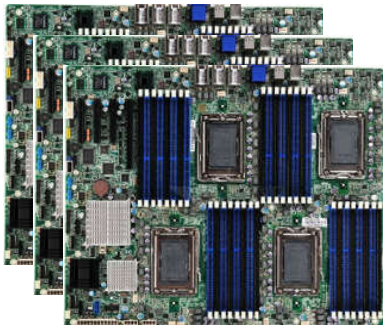
<NUMA>



➢Distributed Memory

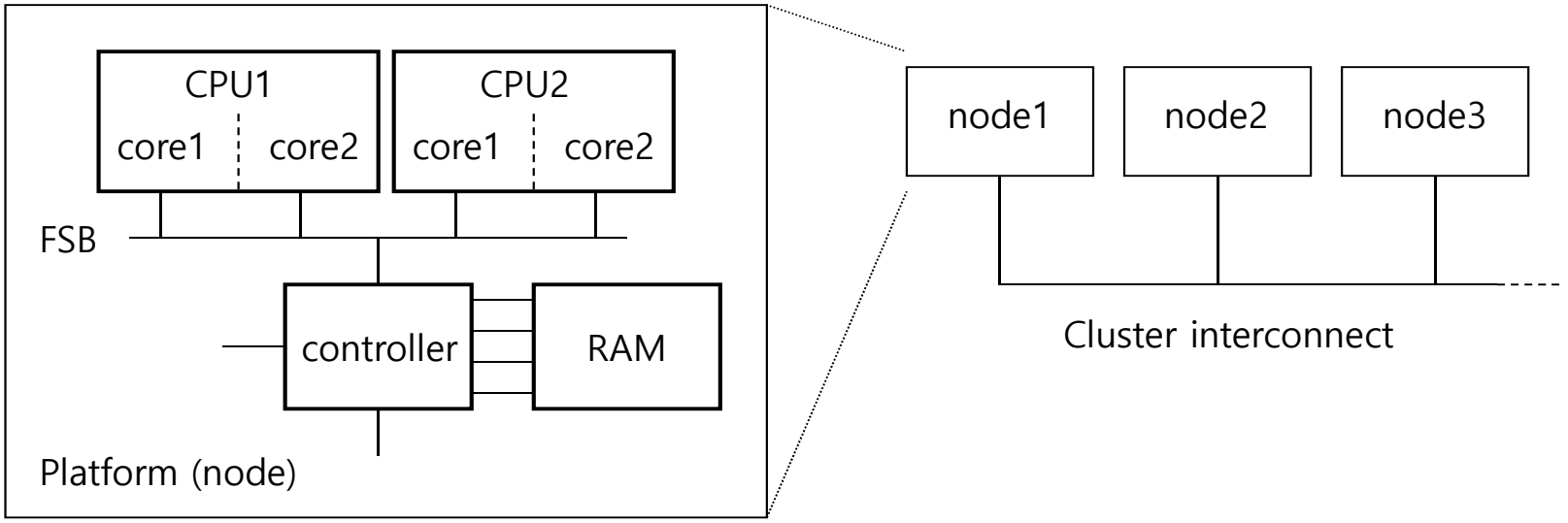






Shared vs Distributed Memory

Multi-core CPUs in clusters – two types of parallelism to consider



Shared memory on each node…

Pthreads
OpenMP
Automatic Parallelization

Distributed memory across cluster

Sockets
PVM
MPI

How To Program A Parallel Program?

➢ **A Single System("Shared Memory")**

- Native Threading Model(standardized, low level)
- Open MP(de-facto standard)
- Automatic Parallelization(compiler does if for you)

➢ **A Cluster of Systems("Distributed Memory")**

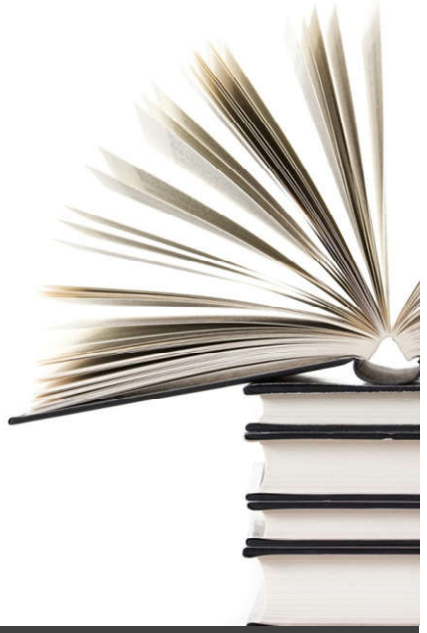
- Sockets(standardized, low level)
- PVM – Parallel Virtual Machine (obsolete)
- MPI – Message Passing Interface (de-facto standard)

Parallel Models Compared

	MPI	Threads	OpenMP*
Portable	✓		✓
Scalable	✓	✓	✓
Performance Oriented	✓		✓
Supports Data Parallel	✓	✓	✓
Incremental Parallelism			✓
High Level			✓
Serial Code Intact			✓
Verifiable Correctness			✓
★ Distributed Memory	✓		

OpenMP Basics I

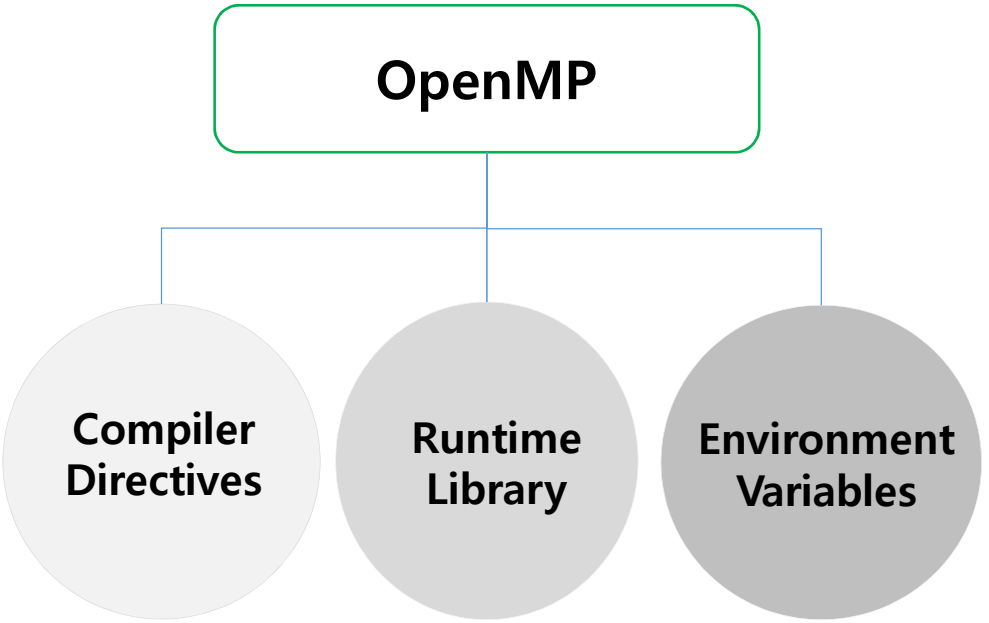
1. Introduction to OpenMP
2. Create Threads
3. Data Scope Attribute



What is OpenMP ?

➢OpenMP is
- De-facto standard API(Application Programming Interface) for multi-thread based shared memory parallel programming
- Not a new programming language
- Notation that can be added to a sequential program in C, C++ and Fortran

➢Consists of
- Compiler Directives
- Runtime Library (Functions)
- Environment Variables



Components of OpenMP (1/2)

➢Compiler Directives

- communicate with the compiler on parallelism

  ex) !$OMP PARALLEL DO

➢Runtime Library (Functions)

- enables the setting and querying of parallel parameters such as number of participating threads and the thread number

  ex) CALL omp_set_num_threads(128)

➢Environmental Variables

- define runtime system parallel parameters such as the number of threads

  ex) export OMP_NUM_THREADS=8

Components of OpenMP (2/2)

Fortran

```
PROGRAM omp_component

      INTEGER omp_get_thread_num

!$OMP PARALLEL
      PRINT *, 'Hello World', omp_get_thread_num()
!$OMP END PARALLEL

END
```

Compiler Directive

Runtime Library

```
$ ifort -openmp –o omp_component.x omp_component.f90
$ export OMP_NUM_THREADS=4
$ ./omp_component.x
```

Environment Variable

C

```
#include <stdio.h>
#include <omp.h>

int main()
{
#pragma omp parallel
{
        printf ("Hello World %d₩n", omp_get_thread_num());
}

        return 0;
}
```

Compiler Directive

Runtime Library

```
$ icc –openmp –o omp_component.x omp_component.c
$ export OMP_NUM_THREADS=4
$ ./omp_component.x
```

Environment Variable

➢OpenMP Syntax

	Fortran	C
Compiler Directive	!$OMP <directive>	#pragma omp <directive>
Runtime Library	omp_	omp_
Environmental Variable	OMP_	OMP_

Creating Threads (1/4)

Fortran	C

PROGRAM create_thread
  IMPLICIT NONE
  INTEGER omp_get_thread_num
 	
!$OMP PARALLEL
  PRINT *, 'Hello World', omp_get_thread_num()
!$OMP END PARALLEL
  print *, ''
 	
  CALL omp_set_num_threads(4)
!$OMP PARALLEL
  PRINT *, 'Hello World', omp_get_thread_num()
!$OMP END PARALLEL
  print*, ''
 	
!$OMP PARALLEL num_threads(2)
  PRINT *, 'Hello World', omp_get_thread_num()
!$OMP END PARALLEL
 	
END
	#include <stdio.h>
	#include <omp.h>
	int main(){
	#pragma omp parallel
	{
	  printf ("Hello World %d₩n", omp_get_thread_num());
	}
	  printf("₩n");
	  omp_set_num_threads(4);
	#pragma omp parallel
	{
	  printf ("Hello World %d₩n", omp_get_thread_num());
	}
	  printf("₩n");
	#pragma omp parallel num_threads(2)
	{
	  printf ("Hello World %d₩n", omp_get_thread_num());
	}
	  return 0;
	}

$ ifort -openmp –o create_thread.x create_thread.f90
$ export OMP_NUM_THREADS=8
$ ./create_thread.x
	$ icc –openmp –o create_thread.x create_thread.c
	$ export OMP_NUM_THREADS=8
	$ ./create_thread.x

Creating Threads (2/4)

➢Parallel Region

Fortran	C

!$OMP PARALLEL
!$OMP END PARALLEL
	#pragma omp parallel
	{
	}

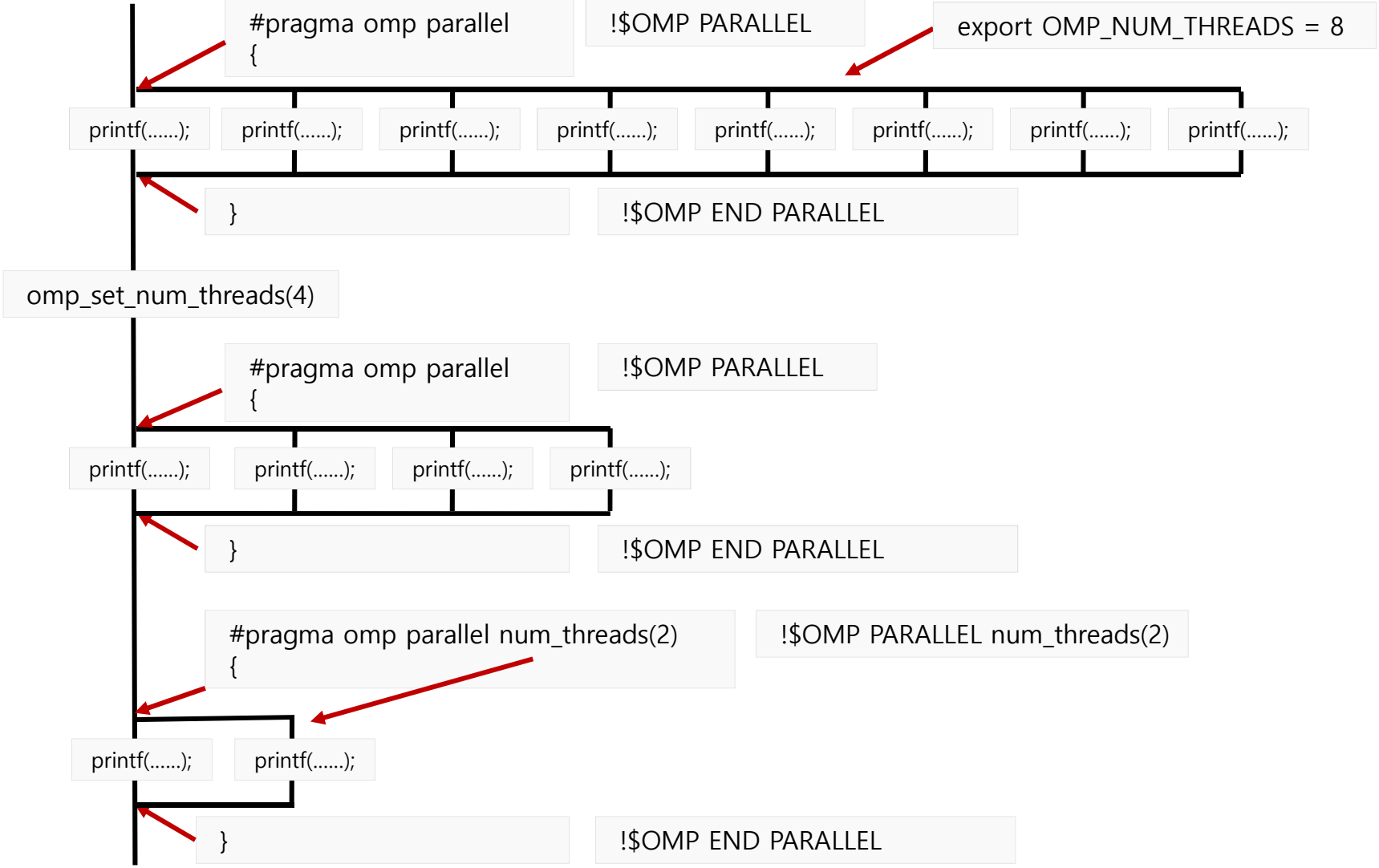
➢Set number of thread

- Environment Variable : export OMP_NUM_THREADS = xxx
- Runtime Library : omp_set_num_threads(xxx)
- Directive : #pragma omp parallel num_threads(xxx)

➢Runtime libraries related to thread

- omp_set_num_threads(integer) : Affects the number of threads
- omp_get_num_threads() : Returns the number of threads in the current team
- omp_get_thread_num() : Returns the ID of the encountering thread
- omp_get_max_threads()
- omp_get_num_procs()

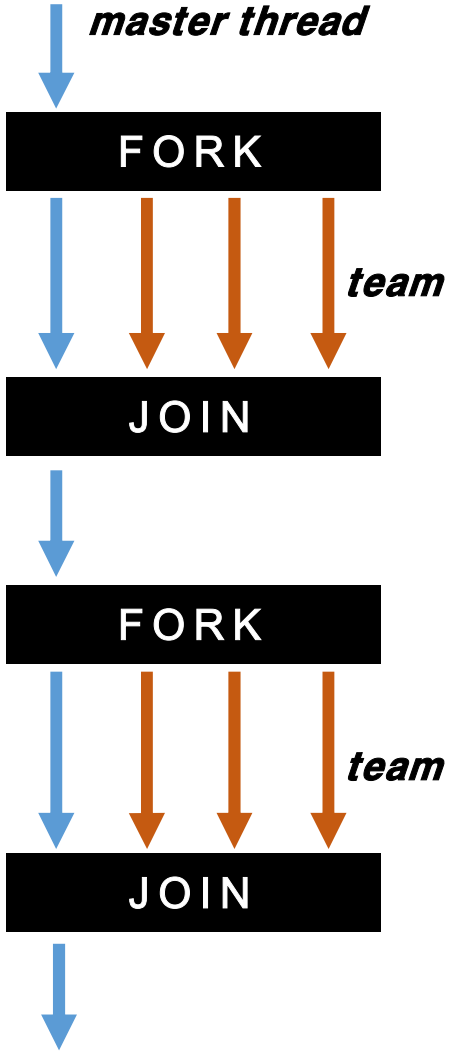
Creating Threads (3/4)



Creating Threads (4/4)

➢OpenMP Programming Model
- Thread-Based
- Fork-Join Model

➢Fork-Join Model
- The master thread spawns a team of threads that joins at the end of the parallel region
- Threads in the same team can collaborate to do work



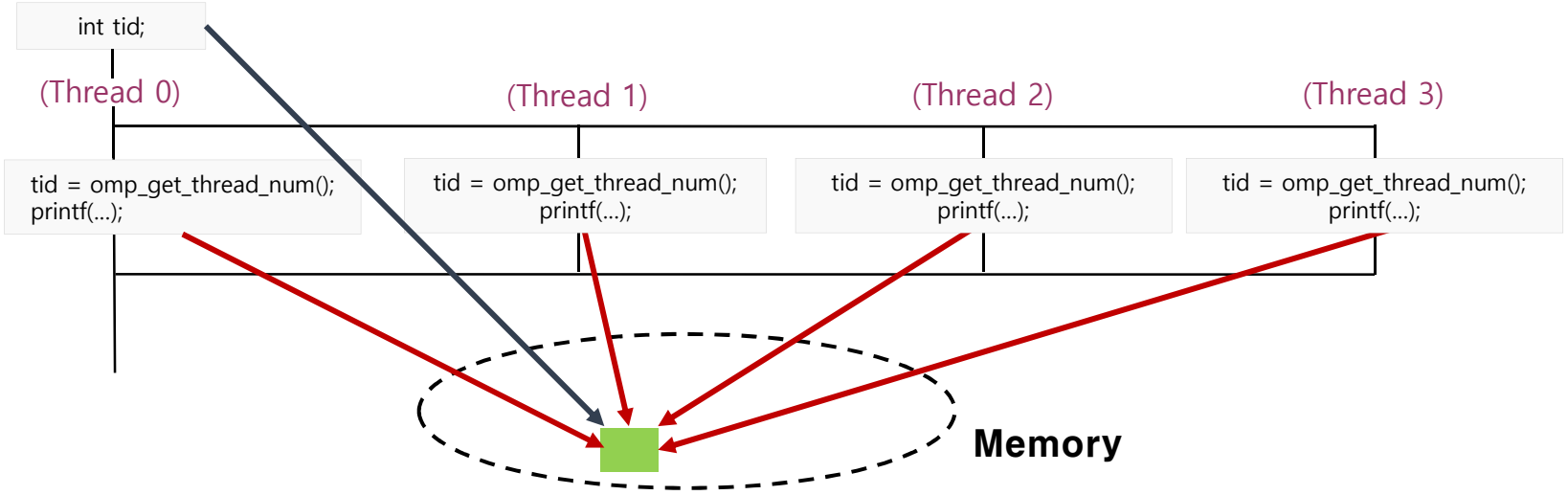
Data Scope Attribute (1/7)

Fortran

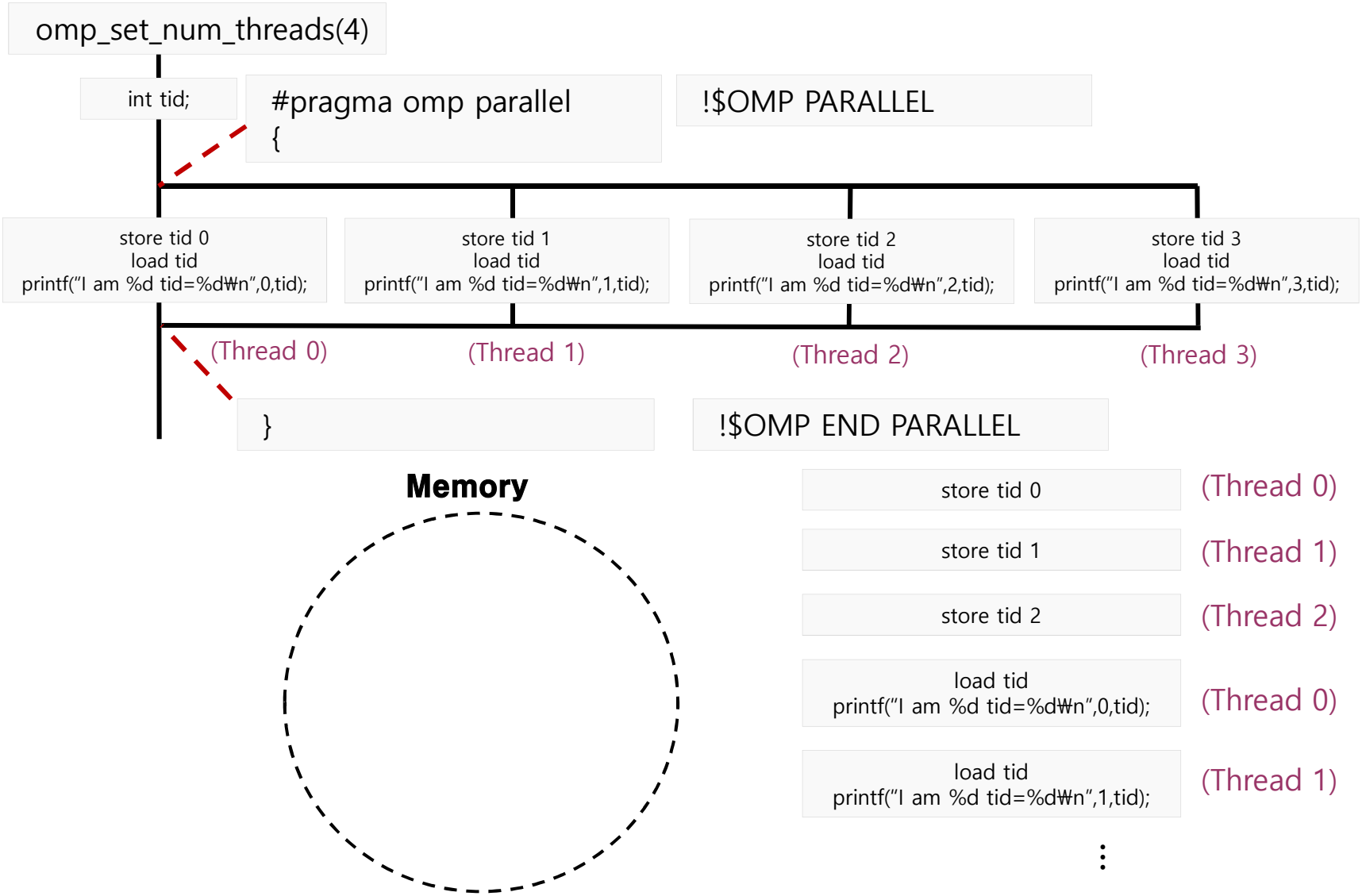
```
PROGRAM hello_wrong

      INTEGER tid, omp_get_thread_num

      CALL omp_set_num_threads(4)
!$OMP PARALLEL
      tid = omp_get_thread_num()
      PRINT *, ' I am ', omp_get_thread_num(), ' tid = ', tid
!$OMP END PARALLEL

END
```

C

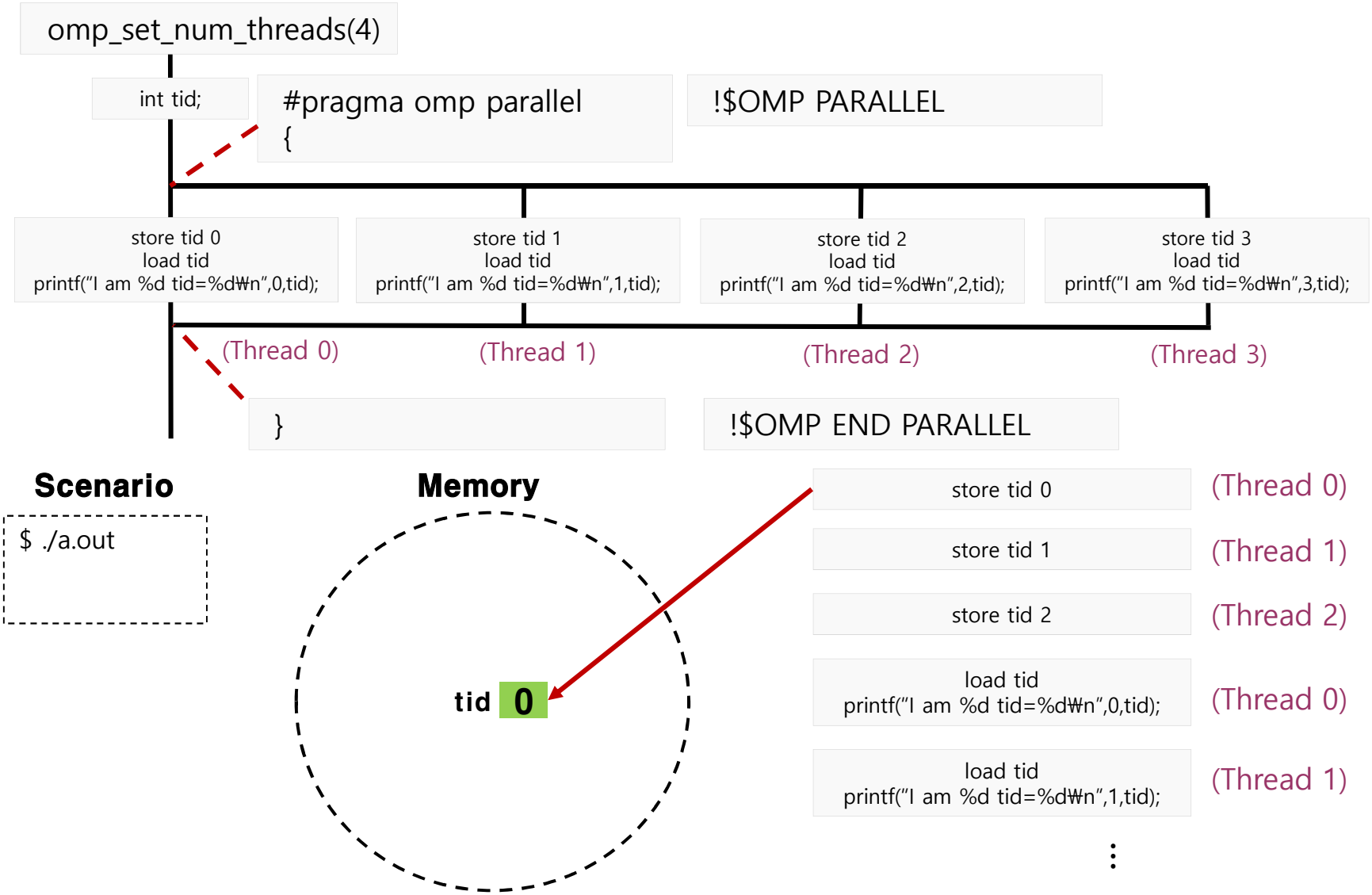
```
#include <stdio.h>
#include <omp.h>
int main()
{
      int tid;
      omp_set_num_threads(4);
#pragma omp parallel
{
      tid = omp_get_thread_num(); //sleep(1);
      printf ("I am %d tid = %d₩n", omp_get_thread_num(), tid);
}
      return 0;
}
```



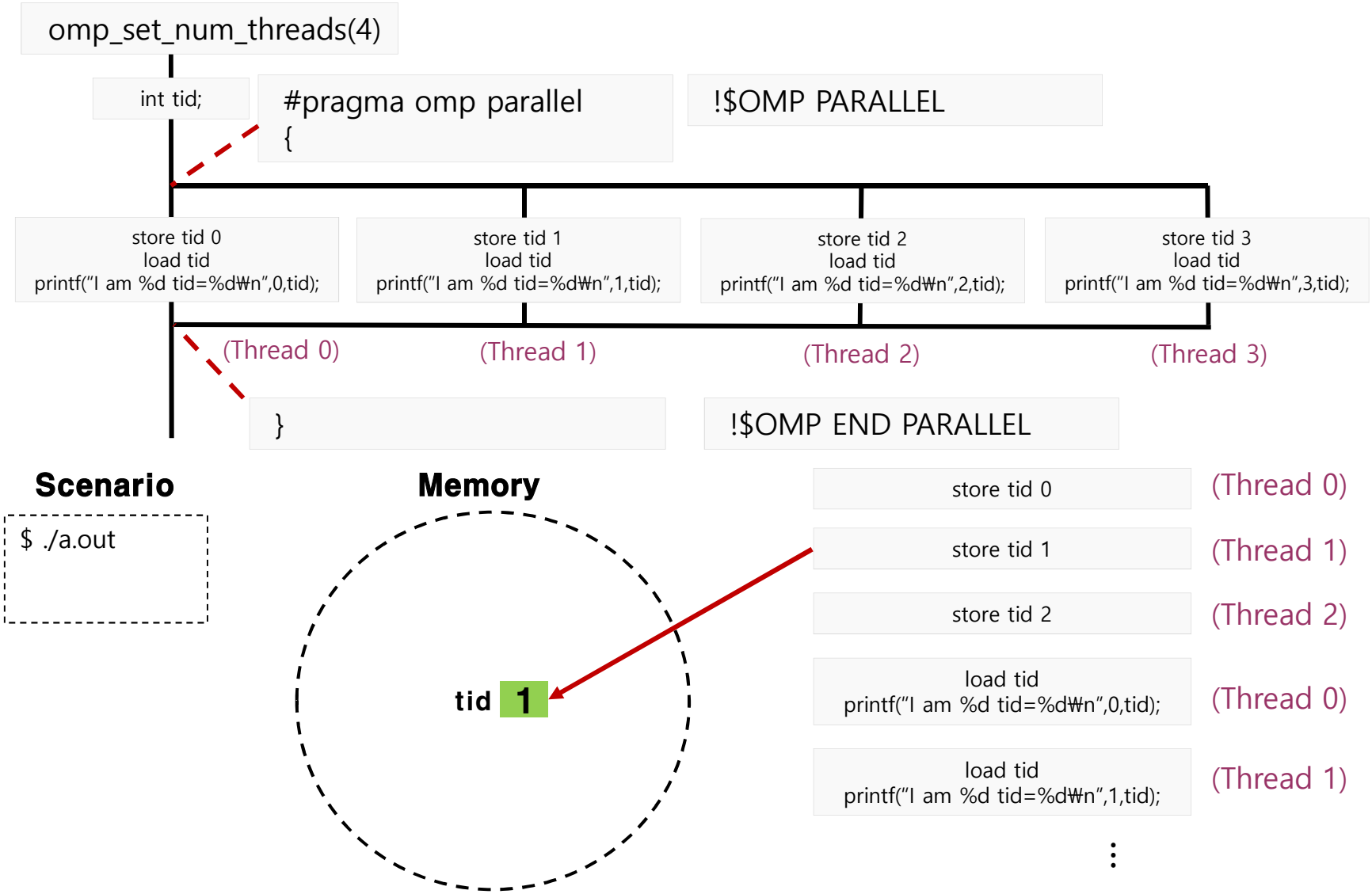
Data Scope Attribute (1/7)



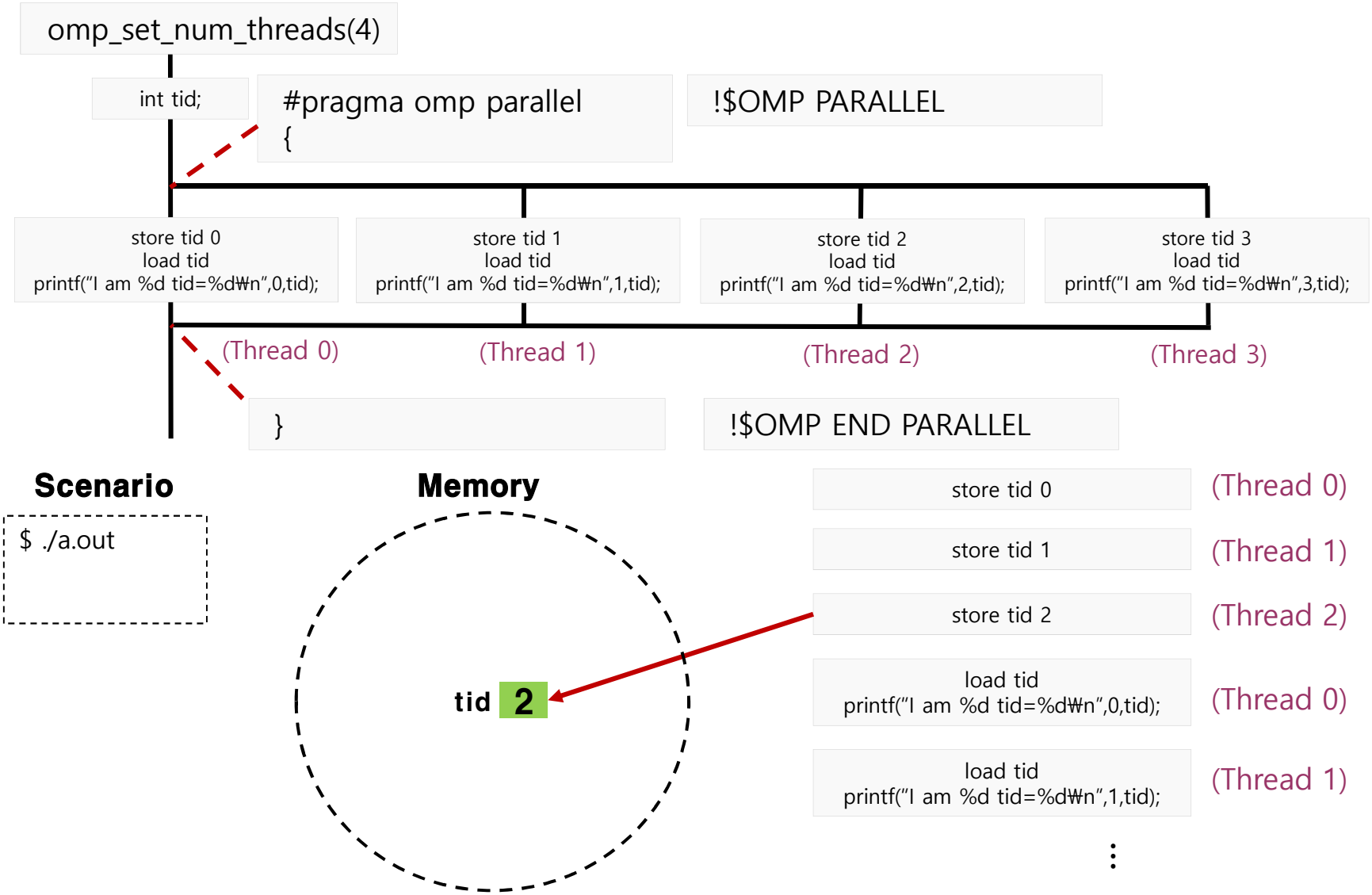
Data Scope Attribute (1/7)



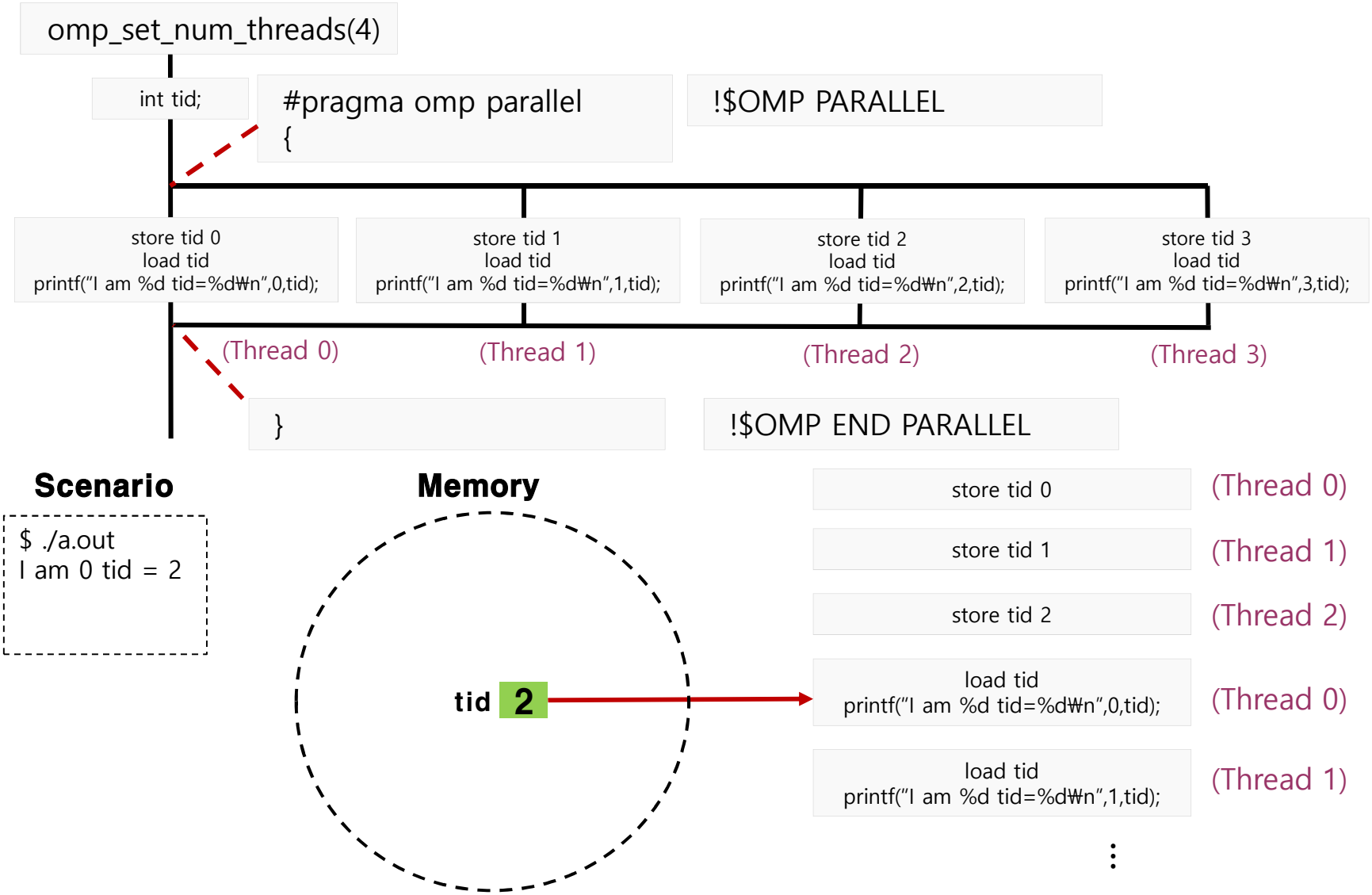
Data Scope Attribute (1/7)



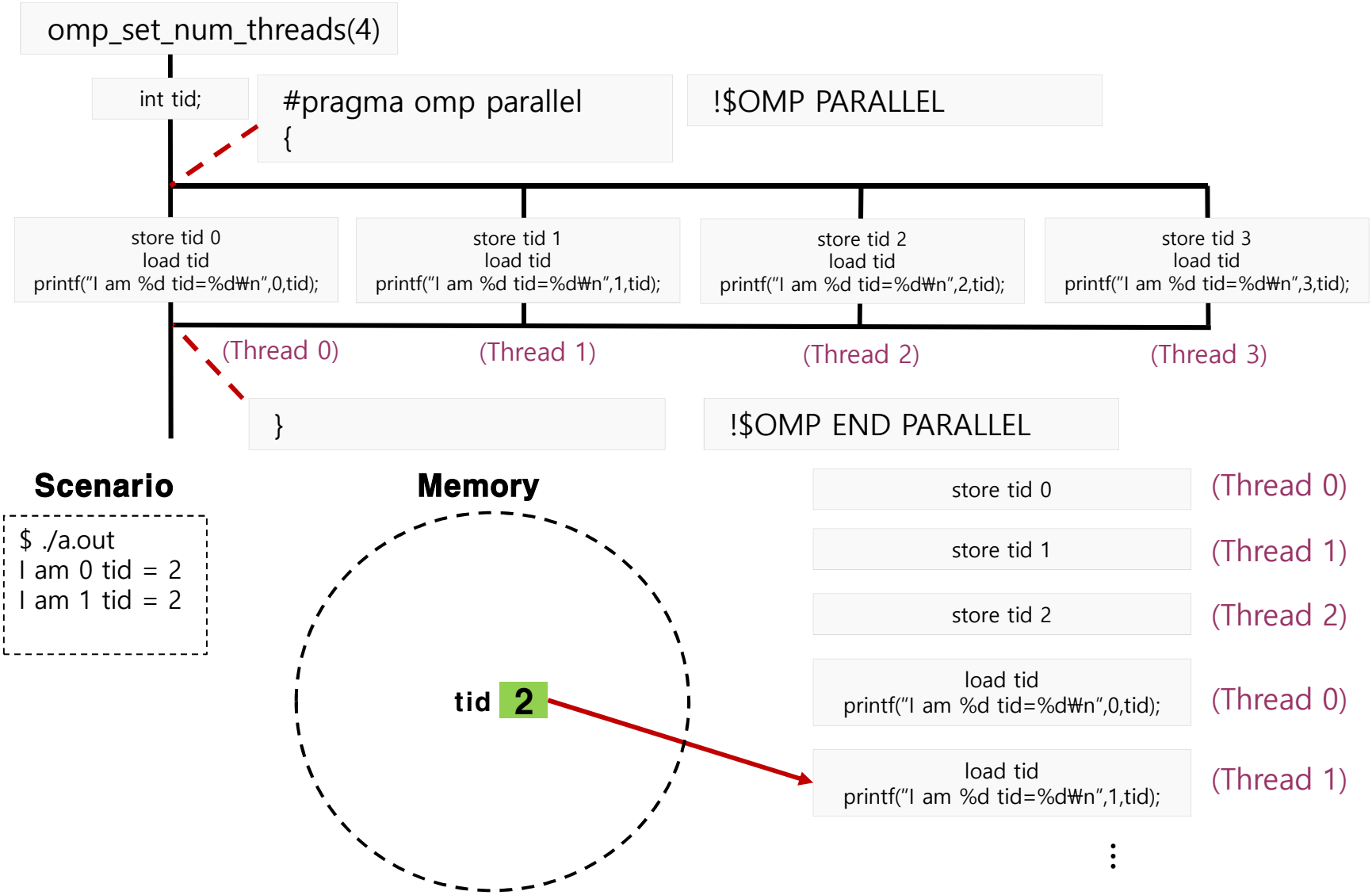
Data Scope Attribute (1/7)



Data Scope Attribute (1/7)



Data Scope Attribute (1/7)



Data Scope Attribute (2/7)

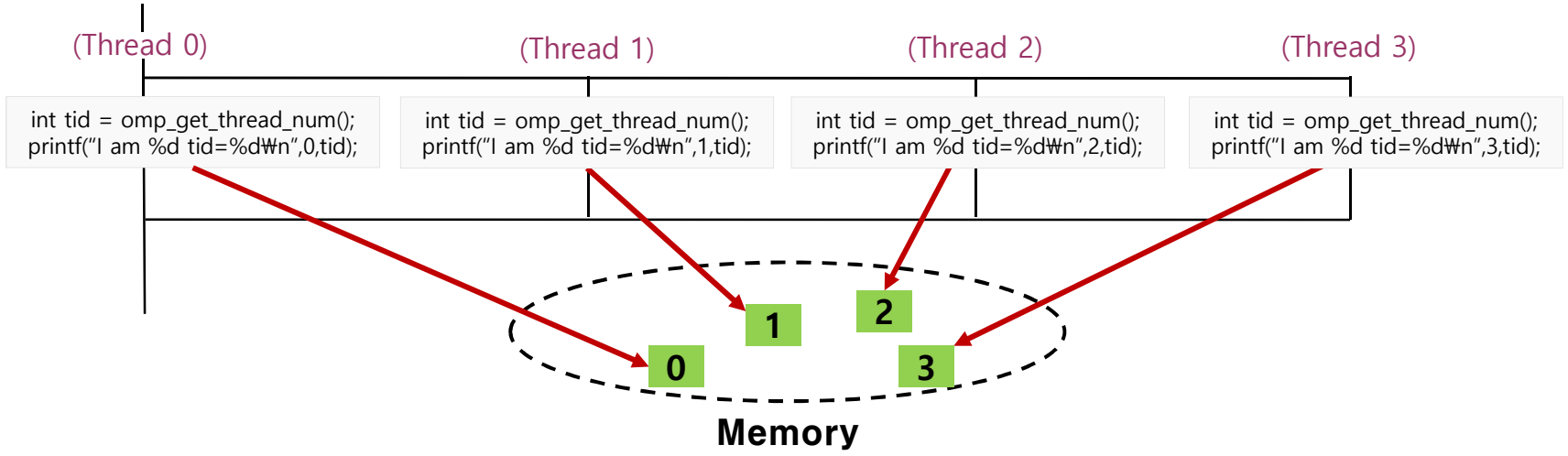
Fortran

```
PROGRAM hello_wrong
      INTEGER omp_get_thread_num
      call omp_set_num_threads(4)

!$OMP PARALLEL
      INTEGER tid
      tid = OMP_GET_THREAD_NUM()
      PRINT *, 'I am',     &
      OMP_GET_THREAD_NUM(), &
            'TID =' tid
!$OMP END PARALLEL
END
```

Not allowed

C

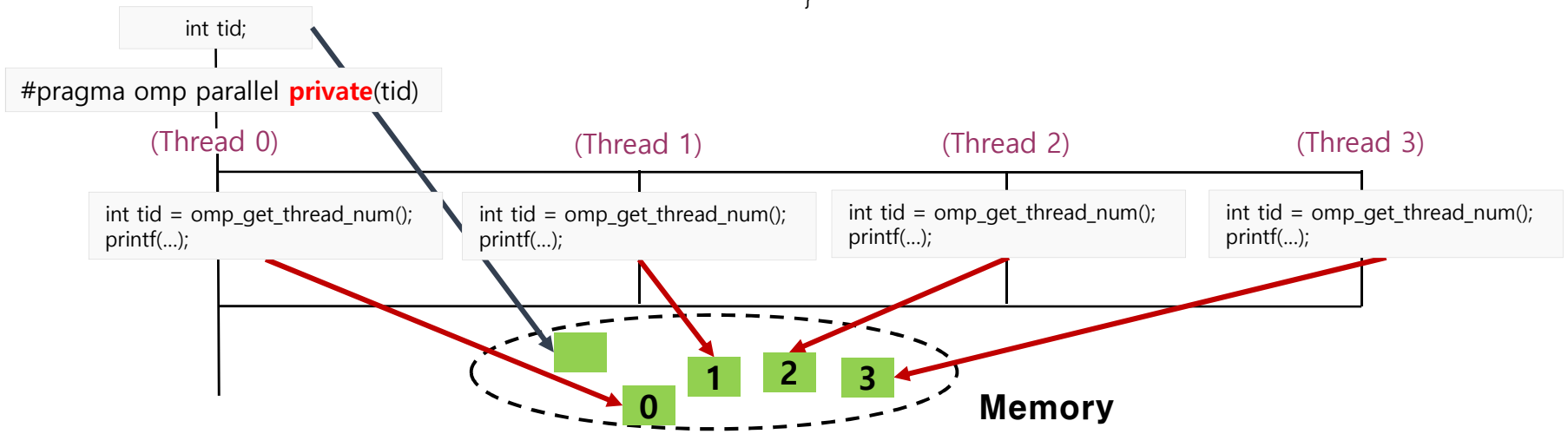
```
#include <stdio.h>
#include <omp.h>
int main(){
omp_set_num_threads(4);
#pragma omp parallel
{
      int tid = omp_get_thread_num();
      printf ("I am %d tid = %d₩n", omp_get_thread_num(), tid);
}
      return 0;
}
```



Data Scope Attribute (3/7)

Fortran	C

PROGRAM hello_right
INTEGER tid, omp_get_thread_num
 	
CALL omp_set_num_threads(4)
!$OMP PARALLEL PRIVATE(tid)
tid = OMP_GET_THREAD_NUM()
PRINT *, 'I am ', OMP_GET_THREAD_NUM(), ' tid = ', tid
!$OMP END PARALLEL
 	
END
	#include <stdio.h>
	#include <omp.h>
	int main(){
	int tid;
	omp_set_num_threads(4);
	#pragma omp parallel private(tid)
	{
	tid = omp_get_thread_num();
	printf ("I am %d tid = %d₩n", omp_get_thread_num(), tid);
	}
	return 0;
	}



Data Scope Attribute (4/7)

Fortran

```
PROGRAM firstprivate

      INTEGER :: i = 10, tid, omp_get_thread_num

      call omp_set_num_threads(4)
!$OMP PARALLEL PRIVATE(tid) FIRSTPRIVATE(i)
      tid = OMP_GET_THREAD_NUM()
      PRINT *, 'tid =' , tid, 'i =', i
      i = 20
!$OMP END PARALLEL
      print *, 'tid = ', tid, 'i =', i

END
```

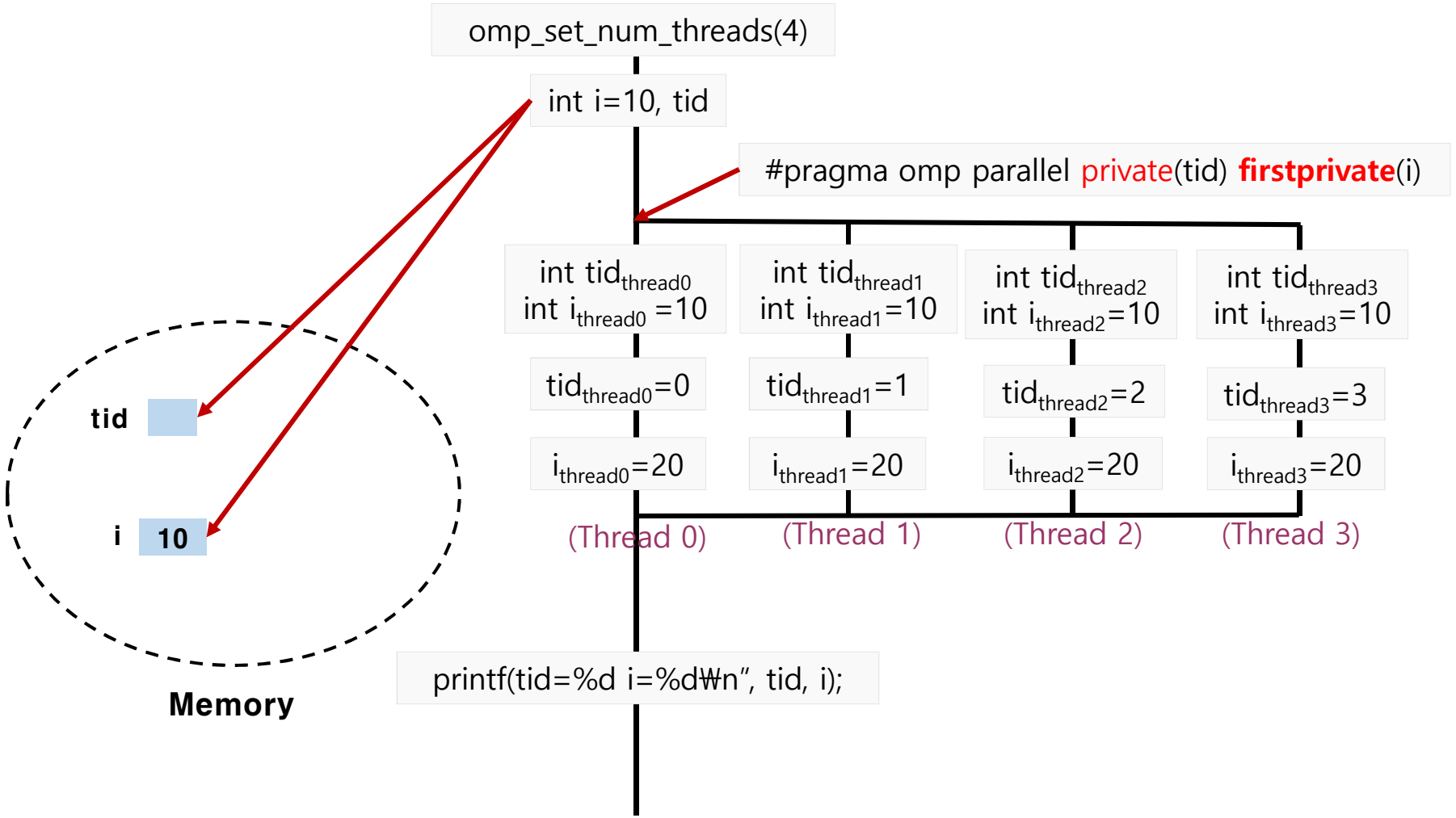
```
$ gfortran -fopenmp -o firstprivate.x firstprivate.f90
$ ./firstprivate.x
```

C

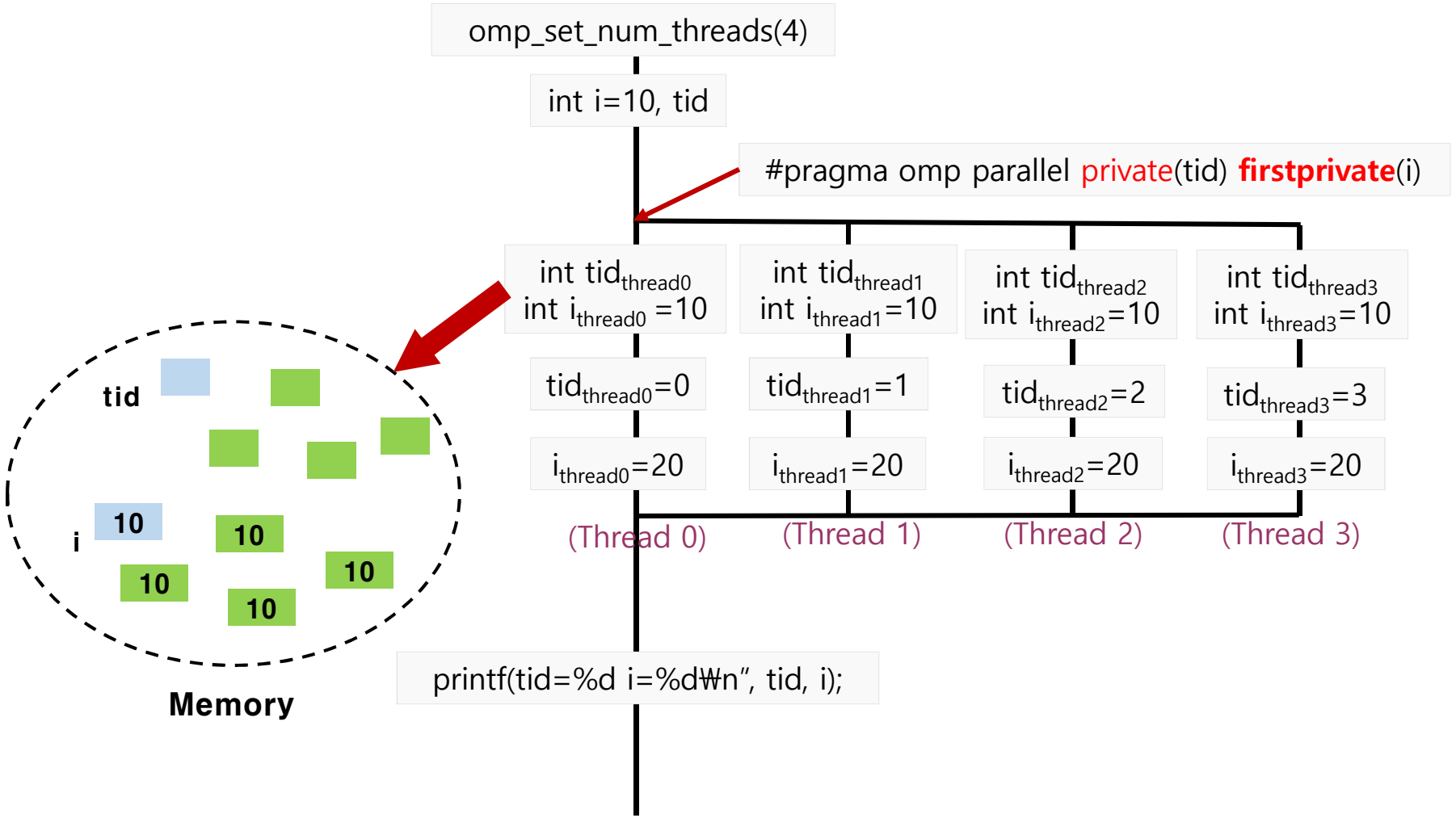
```
#include <stdio.h>
#include <omp.h>

int main()
{
      int i = 10, tid;

      omp_set_num_threads(4);
#pragma omp parallel private(tid) firstprivate(i)
{
      tid = omp_get_thread_num();
      printf ("tid = %d i = %d₩n", tid, i);
      i = 20;
}
      printf("tid = %d i=%d₩n", tid, i);

      return 0;
}
```

```
$ gcc -fopenmp -o firstprivate.x firstprivate.c
$ ./firstprivate.x
```

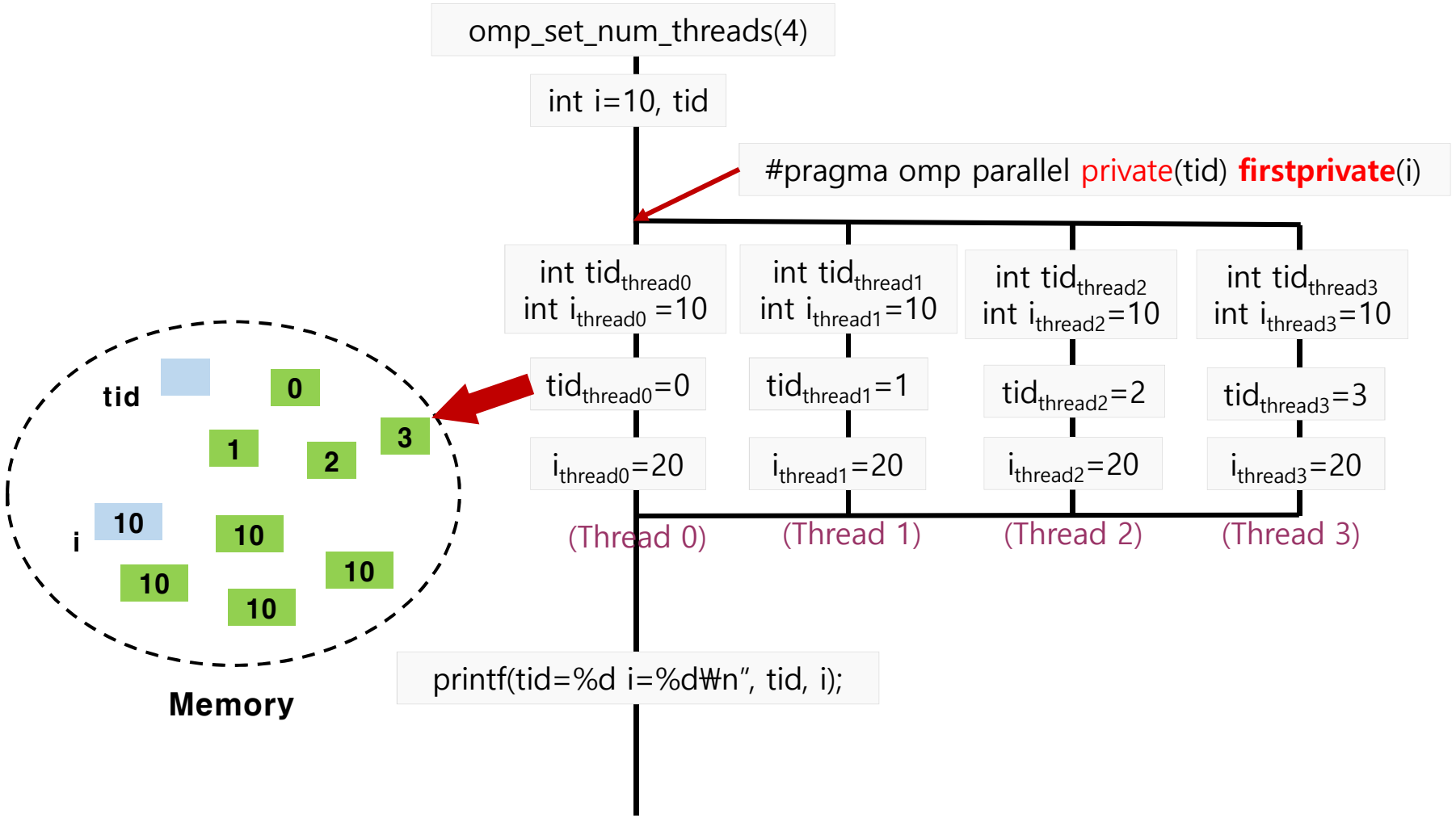
Data Scope Attribute (5/7)



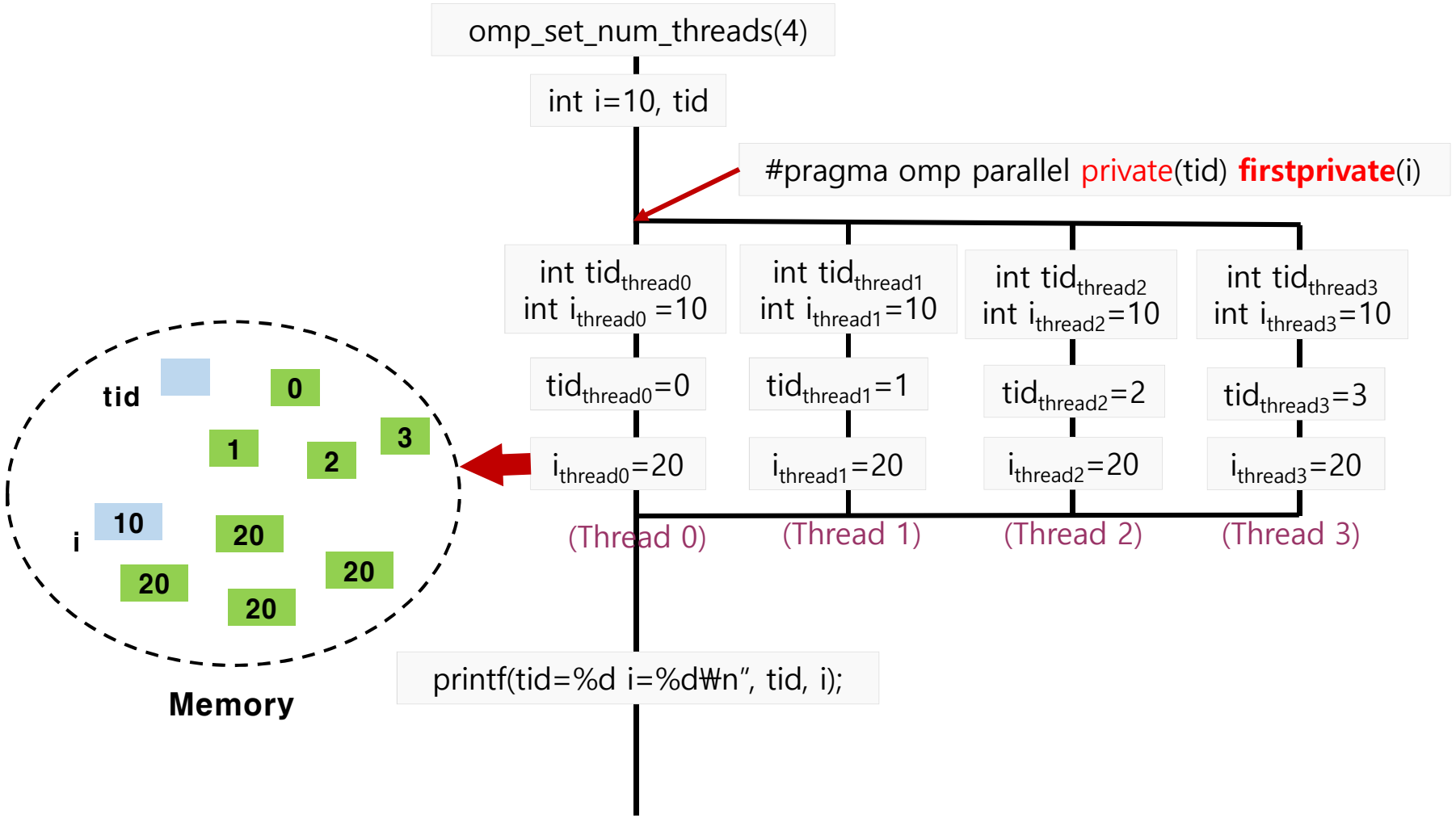
Data Scope Attribute (5/7)



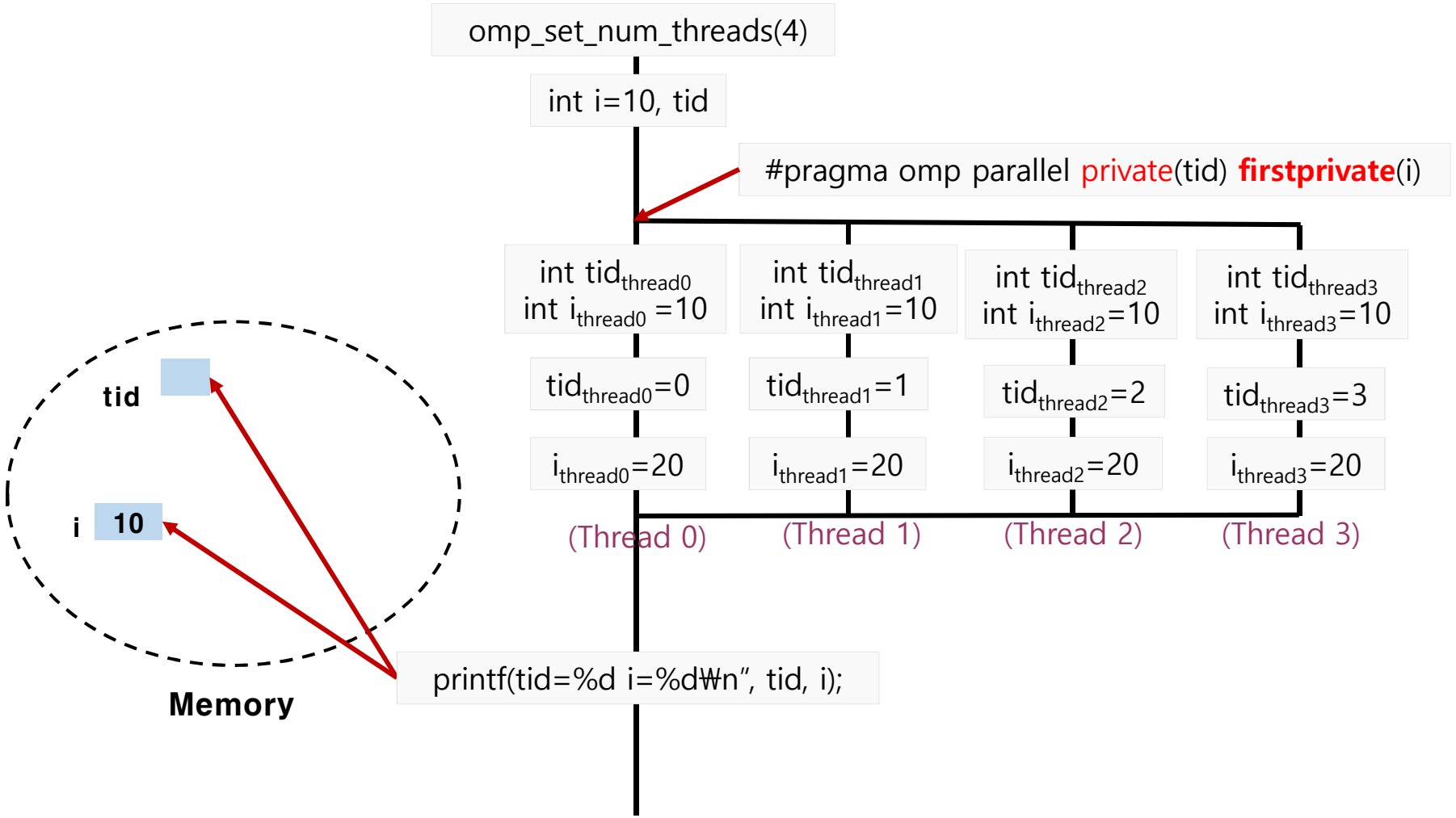
Data Scope Attribute (5/7)



Data Scope Attribute (5/7)



Data Scope Attribute (5/7)



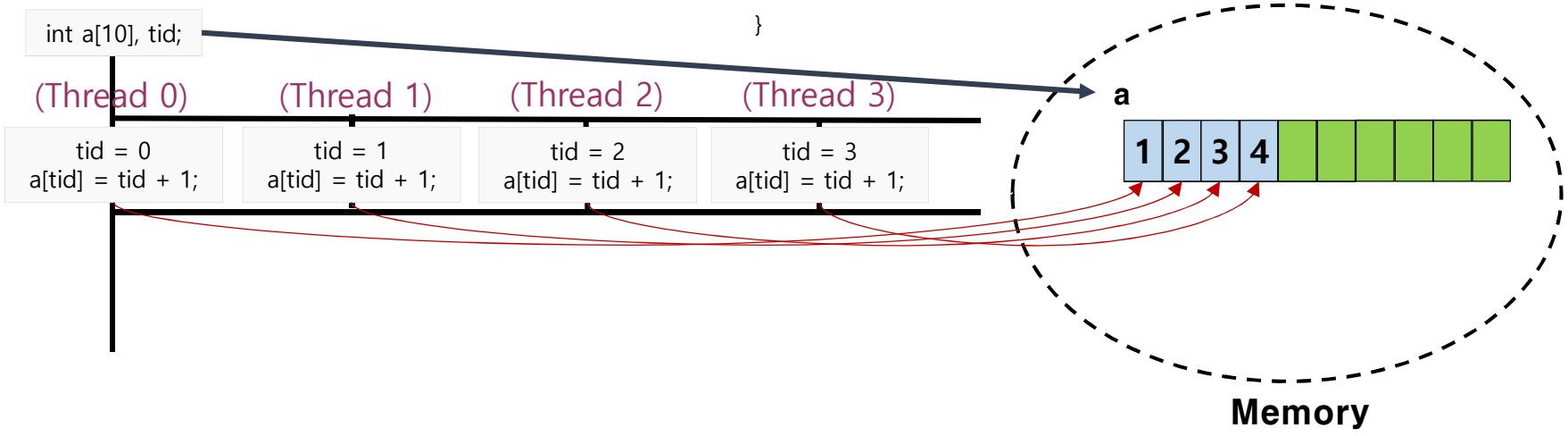
Data Scope Attribute (6/7)

Fortran

```
PROGRAM shared
      INTEGER a(0:9), tid, i, omp_get_thread_num

      CALL omp_set_num_threads(4)

!$OMP PARALLEL SHARED(a) PRIVATE(tid)
      tid = OMP_GET_THREAD_NUM()
      a(tid) = tid + 1
!$OMP END PARALLEL

      DO i=0, 3
         print *, 'a(', i, ') =', a(i)
      END DO

END
```

C

```
#include <stdio.h>
#include <omp.h>
int main(){
        int a[10], tid, i;
        omp_set_num_threads(4);
#pragma omp parallel shared(a) private(tid)
{
        tid = omp_get_thread_num();
        a[tid] = tid + 1;
}
        for(i=0; i<4; i++)
        printf("a[%d] = %d₩n", i, a[i]);
        return 0;
}
```



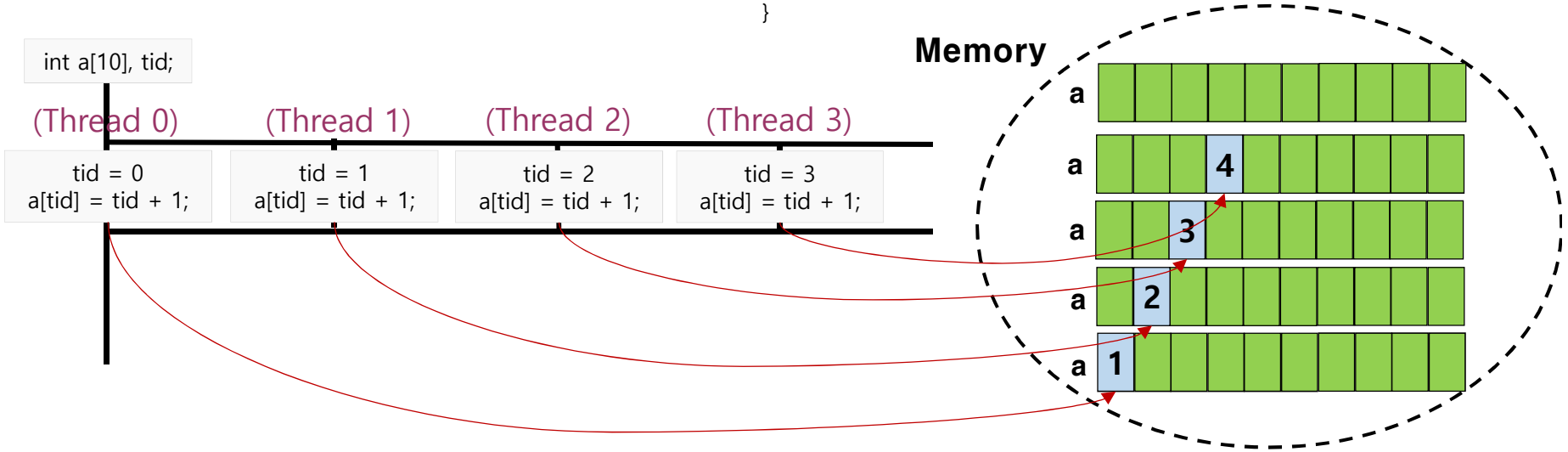
Data Scope Attribute (6/7)

Fortran

```
PROGRAM data_scope_private_array
        INTEGER a(0:9), tid, i, omp_get_thread_num
        CALL omp_set_num_threads(4)
!$OMP PARALLEL PRIVATE(a) PRIVATE(tid)
        tid = OMP_GET_THREAD_NUM()
        a(tid) = tid + 1
!$OMP END PARALLEL
        DO i=0, 3
                PRINT *, 'a(', i, ') =', a(i)
        END DO
END
```

C

```
#include <stdio.h>
#include <omp.h>
int main(){
        int a[10], tid, i;
        omp_set_num_threads(4);
#pragma omp parallel private(a) private(tid)
{
        tid = omp_get_thread_num();      a[tid] = tid + 1;
}
        for(i=0; i<4; i++)
                printf("a[%d] = %d₩n", i, a[i]);
        return 0;
}
```



Advanced Lab: PI (Numerical Integration)

➢ **<Problem>**

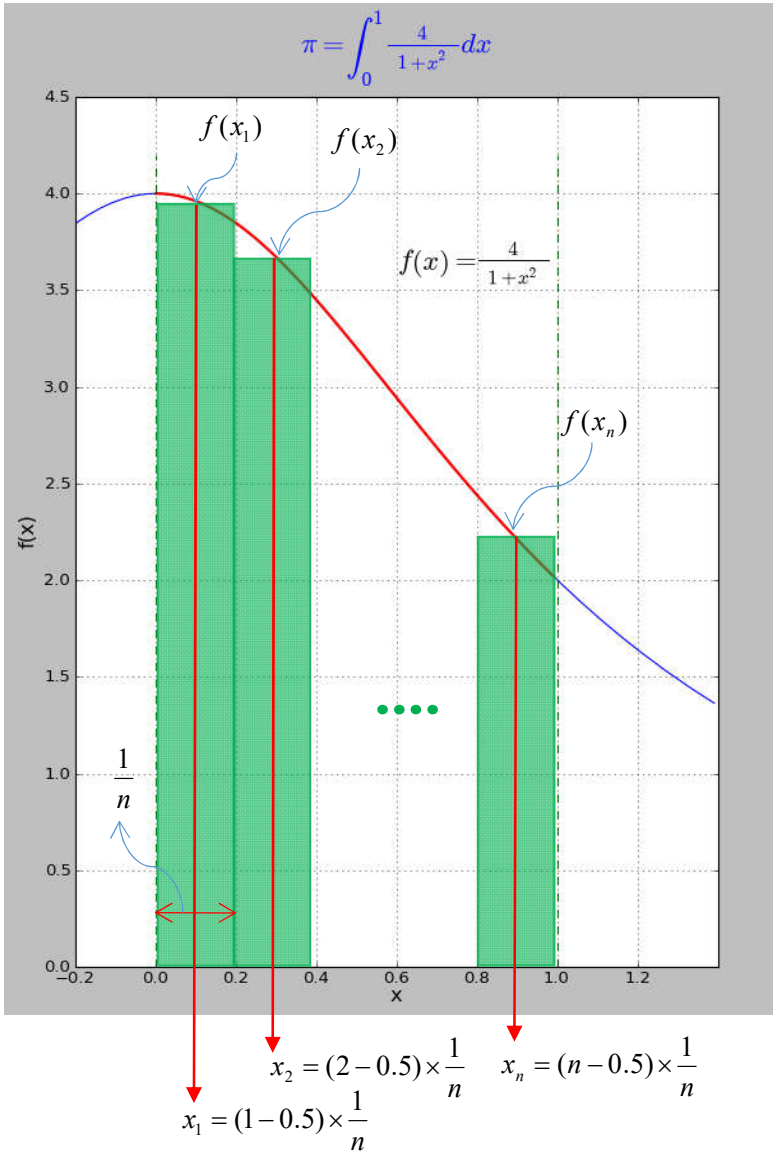
- Get PI using Numerical integration

$$\int_0^1 \frac{4.0}{(1+x^2)} dx = \pi$$

➢ **<Requirement>**

- Point to point communication

$$\pi \approx \sum_{i=1}^{n} \frac{4}{1+((i-0.5)\times\frac{1}{n})^2} \times \frac{1}{n}$$



Advanced Lab: PI Numerical Integration :serial - C

```
#include <stdio.h>
#include <math.h>
#define num_steps 1000000000

void main(int argc, char *argv[]) {
  double sum, step, x, pi;
  double t1, t2;
  int i;

  sum=0.0;
  step=1./(double)num_steps;

  for(i=1; i<num_steps; i++){
     x = (i-0.5)*step;
     sum = sum + 4.0/(1.0+x*x);
  }

  pi = step*sum;
  printf(" numerical  pi = %.15f \n", pi);
  printf(" analytical pi = %.15f \n", acos(-1.0));
  printf(" Error = %E \n", fabs(acos(-1.0)-pi));
}
```

Advanced Lab: PI Numerical Integration :Serial - Fortran

```
integer, parameter:: num_steps=1000000000
real(8) sum, step, x, pi;

sum=0.0
step=1./dble(num_steps)

do i=1, num_steps
   x = (i-0.5)*step
   sum = sum + 4.0/(1.0+x*x)
enddo

pi = step*sum
print*, "numerical  pi = ", pi
print*, "analytical pi = ", dacos(-1.d0)
print*, " Error = ", dabs(dacos(-1.d0)-pi)

end
```

Parallel: C

```
#pragma omp parallel private(id,x)
{
        id = omp_get_thread_num();
        sum[id]=0.0;
        for (i=id; i< num_steps; i=i+NUM_THREADS){
         x = (i+0.5)*step;
         sum[id] += 4.0/(1.0+x*x);
    }
}
for(i=0, pi=0.0;i<NUM_THREADS;i++)
         pi += sum[i] * step;
```

Parallel: Fortran

```
!$OMP PARALLEL PRIVATE(ID,X)
  ID = OMP_GET_THREAD_NUM()
     SUM(ID)=0.0
  DO I = ID, NUM_STEPS-1, NUM_THREADS
          X = (I+0.5)*STEP
      SUM(ID) = SUM(ID)+ 4.0/(1.0+X*X)
  ENDDO
!$OMP END PARALLEL
     PI=0.0
     DO I=0, NUM_THREADS-1
        PI = PI + SUM(I)*STEP
     ENDDO
```

Parallel Loops

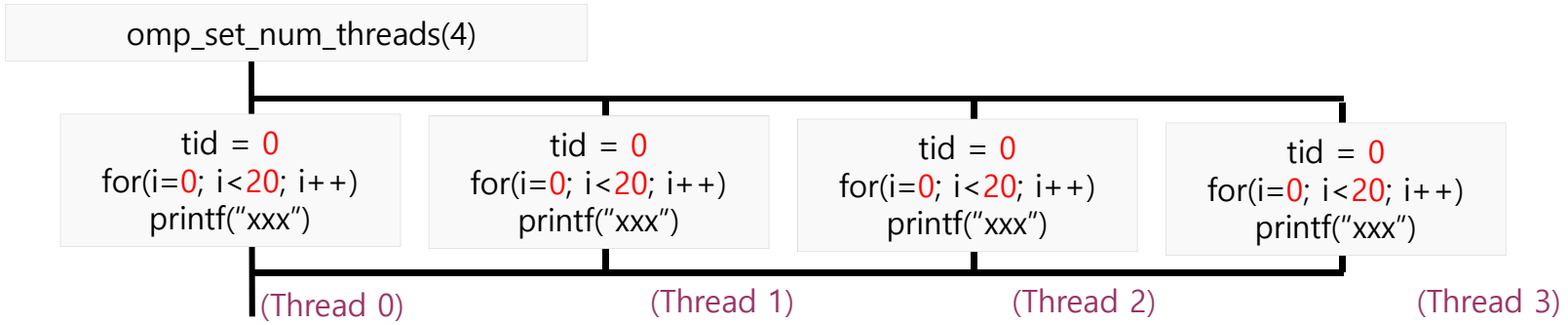
Fortran	C

PROGRAM parallel_for
INTEGER, PARAMETER :: N=20
INTEGER :: tid, i, omp_get_thread_num
 	
CALL omp_set_num_threads(4)
!$OMP PARALLEL private(tid) !!! i : shared??
tid = omp_get_thread_num()
 	
!$OMP DO
DO i=0, N-1
PRINT *, 'Hello World', tid, i
END DO
!$OMP END DO !!![optional]
!$OMP END PARALLEL
 	
END
	#include <stdio.h>
	#include <omp.h>
	#define N 20
	
	int main()
	{
	int tid, i;
	
	omp_set_num_threads(4);
	#pragma omp parallel private(tid) // i : shared?
	{
	tid = omp_get_thread_num();
	#pragma omp for
	for(i=0; i<N; i++)
	printf("Hello World %d %d₩n", tid, i);
	}
	return 0;
	}

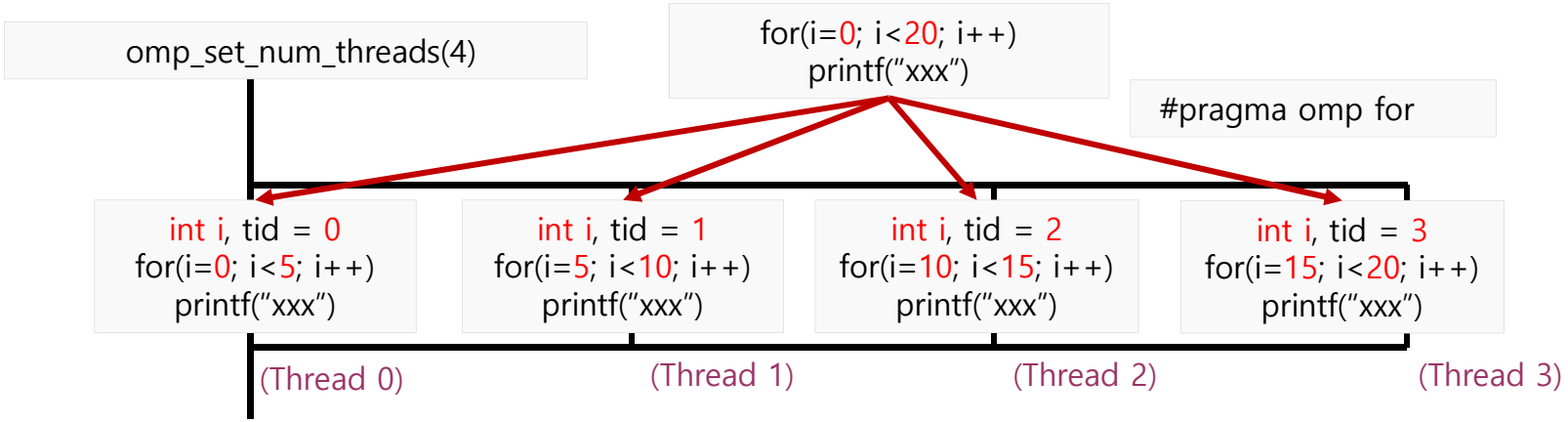
$ gfortran -fopenmp -o parallel_for.x parallel_for.f90 $ ./parallel_for.x	$ gcc -fopenmp -o parallel_for.x parallel_for.c $ ./parallel_for.x

Parallel Loops

➢Without #pragma omp for



➢With #pragma omp for



Parallel Loops

Fortran	C

!$OMP PARALLEL private(tid)
tid = omp_get_thread_num()
DO i=0, N-1
print *, "Hello World", tid, i
END DO
!$OMP END PARALLEL
	#pragma omp parallel private(tid)
	{
	tid = omp_get_thread_num();
	for(i=0; i<N; i++)
	printf("Hello World %d %d₩n", tid, i);
	}

!$OMP PARALLEL private(tid)
tid = omp_get_thread_num()
!$OMP DO
DO i=0, N-1
print *, "Hello World", tid, i
END DO
!$OMP END DO [optional]
!$OMP END PARALLEL
	#pragma omp parallel private(tid)
	{
	tid = omp_get_thread_num();
	#pragma omp for
	for(i=0; i<N; i++)
	printf("Hello World %d %d₩n", tid, i);
	}

➢Work Sharing

- Distribution of work across threads
- **Do/for**, Sections/section, Single, Task construct, etc.
- cf) SPMD (Single Program Multiple Data), WORKSHARE clause in Fortran

Parallel Loops

Fortran	C

!$OMP PARALLEL
!$OMP DO
DO i=0, N-1
print *, "Hello World", i
END DO
!$OMP END DO
!$OMP END PARALLEL
	#pragma omp parallel
	{
	#pragma omp for
	for(i=0; i<N; i++)
	printf("Hello World %d ₩n", i);
	}

!$OMP PARALLEL DO
DO i=0, N-1
print *, "Hello World", i
END DO
!$OMP END PARALLEL DO
	#pragma omp parallel for
	for(i=0; i<N; i++)
	printf("Hello World %d %d₩n", i);

➢ **OpenMP shortcut**

- Combined parallel/worksharing construct

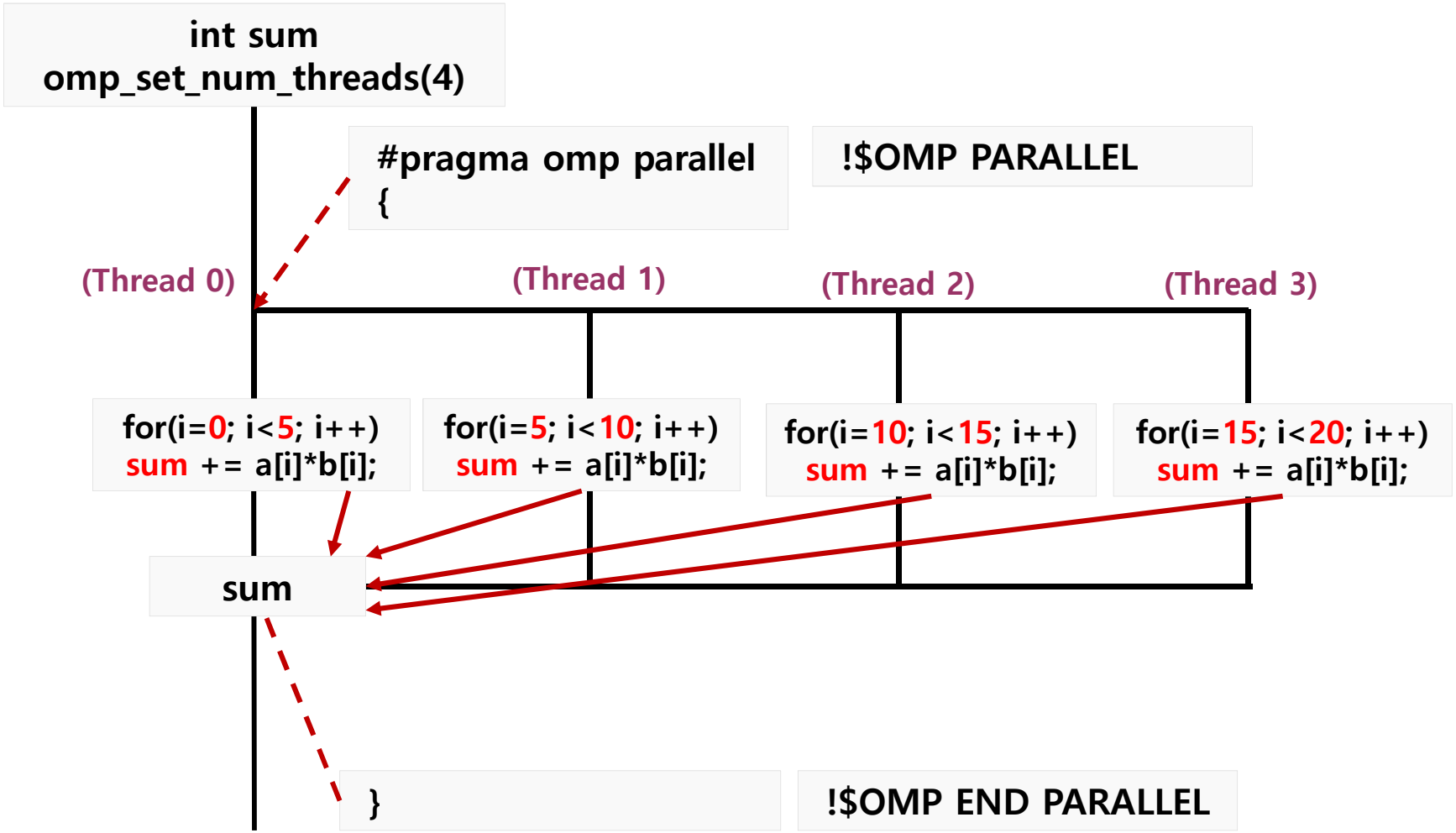
Parallel Do: Fortran

```
!$OMP PARALLEL PRIVATE(ID)
      ID = OMP_GET_THREAD_NUM()
      SUM(ID)=0.0
      !$OMP DO PRIVATE(X)
        DO I = 0, NUM_STEPS-1
          X = (I+0.5)*STEP
          SUM(ID) = SUM(ID) + 4.0/(1.0+X*X)
        ENDDO
      !$OMP END DO
!$OMP END PARALLEL
      PI=0.0
      DO I=0, NUM_THREADS-1
        PI = PI + SUM(I)*STEP
      ENDDO
```

Parallel For: C

```
#pragma omp parallel private(id)
{
    id = omp_get_thread_num();
    sum[id] = 0.0;
    #pragma omp for private(x)
    for (i=0; i<num_steps; i++){
        x = (i+0.5)*step;
        sum[id] += 4.0/(1.0+x*x);
    }
}
for(i=0, pi=0.0; i<NUM_THREADS; i++) pi += sum[i] * step;
```

Synchronization



Synchronization

Fortran		C	

!$OMP PARALLEL
DO i=0, 99
!$OMP CRITICAL(n1)
CALL sub(a,b)
!$OMP END CRITICAL
END DO
!$OMP END PARALLEL
	!$OMP PARALLEL
	DO i=0, 99
	!$OMP ATOMIC
	a = a + b
	END DO
	!$OMP END PARALLEL
		#pragma omp parallel
		{
		for (i=0; i<100; i++) {
		#pragma omp critical(n1)
		func1(a,b);
		}
		}
			#pragma omp parallel
			{
			for (i=0; i<100; i++) {
			#pragma omp atomic
			a += b;
			}
			}

- **Synchronization**
  - Imposing order constraints and protecting access to shared data
- **High level synchronization**
  - **critical, atomic, barrier**, ordered
- **Low level synchronization**
  - flush, locks
- **critical vs atomic**
  - critical : Only one thread at a time can enter a critical region, distinguished by name
  - atomic : Only applies to the update of a memory location
    Like mini-critical section

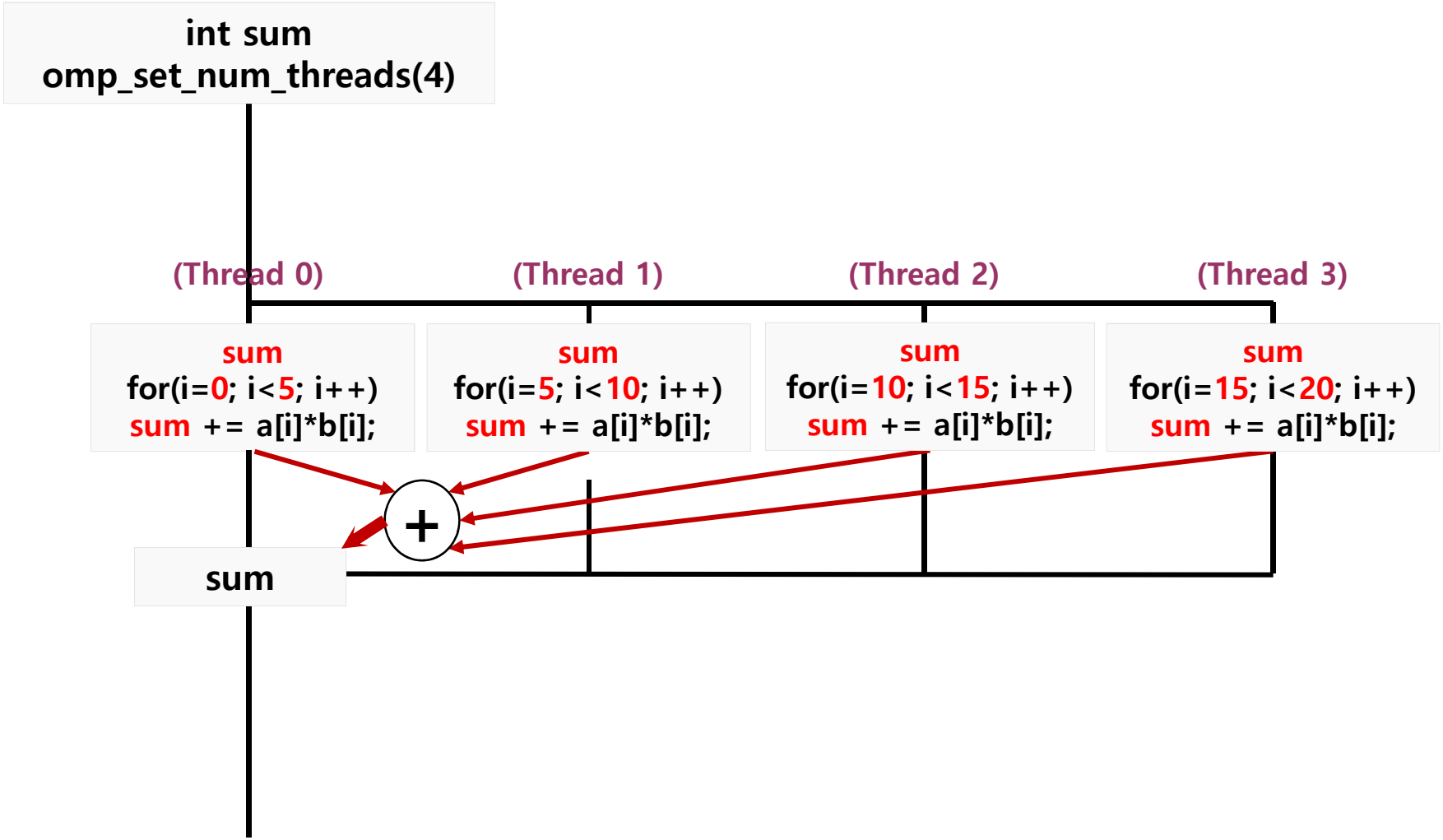
Critical: Fortran

```
!$OMP PARALLEL PRIVATE(ID, X, SUM)
      ID = OMP_GET_THREAD_NUM()
      SUM=0.0
      DO I = ID, NUM_STEPS-1, NUM_THREADS
        X = (I+0.5)*STEP
        SUM= SUM + 4.0/(1.0+X*X)
      ENDDO
      !$OMP CRITICAL
            PI = PI + SUM*STEP
      !$OMP END CRITICAL
!$OMP END PARALLEL
```

Critical: C

```
#pragma omp parallel private (id, x, sum)
{
        int id;
        id = omp_get_thread_num();
        sum=0.0;
        for (i=id; i< num_steps; i=i+NUM_THREADS)
        {
              x = (i+0.5)*step;
              sum += 4.0/(1.0+x*x);
        }
        #pragma omp critical
        {
          pi += sum*step;
        }
}
```

Reduction


int sum
omp_set_num_threads(4)
(Thread 0)
(Thread 1)
(Thread 2)
(Thread 3)
sum
for(i=0; i<5; i++)
sum += a[i]*b[i];
sum
for(i=5; i<10; i++)
sum += a[i]*b[i];
sum
for(i=10; i<15; i++)
sum += a[i]*b[i];
sum
for(i=15; i<20; i++)
sum += a[i]*b[i];
+
sum

Reduction

➢Reduction Operators : Fortran

Operator	Data Types	Initial Value
+	integer, floating point (complex or real)	0
*	integer, floating point (complex or real)	1
-	integer, floating point (complex or real)	0
.AND.	logical	.TRUE.
.OR.	logical	.FALSE.
.EQV.	logical	.TRUE.
.NEQV.	logical	.FALSE.
MAX	integer, floating point (real only)	가능한 최소값
MIN	integer, floating point (real only)	가능한 최대값
IAND	integer	all bits on
IOR	integer	0
IEOR	integer	0

Reduction

➢Reduction Operators : C

Operator	Data Types	Initial Value
+	integer, floating point	0
*	integer, floating point	1
-	integer, floating point	0
&	integer	all bits on
|	integer	0
^	integer	0
&&	integer	1
||	integer	0

Reduction: Fortran

```
      integer num_steps
      real(8) step, x, pi, sum
      num_steps = 500000000
      step = 1.0/real(num_steps)
!$omp parallel do reduction(+:sum) private(x)
      do i = 1, num_steps
        x = (i-0.5)*step
        sum = sum + 4.0/(1.0+x*x)
      enddo
      pi = sum*step
      print *, 'computed pi = ', pi
      print *, 'exact pi =     ',
       '   3.14159_26535_89793_23846_26433_83279_50288_41971_...'

      end
```

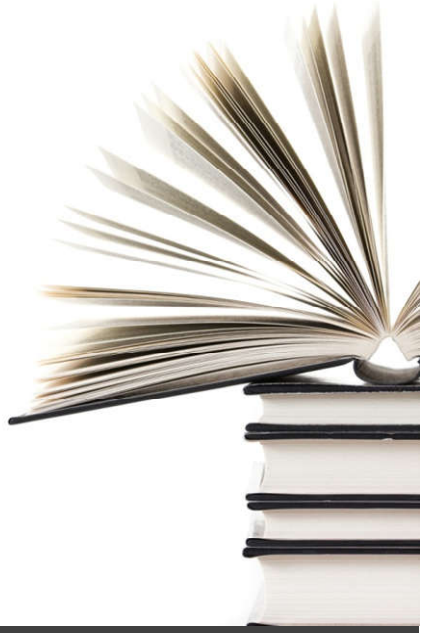
Reduction: C

```
#include <omp.h>
static long num_steps = 500000000;
double step;
void main ()
{     int i;
          double x, pi, sum = 0.0;
      step = 1.0/(double) num_steps;
#pragma omp parallel for reduction(+:sum) private(x)
      for (i=1;i<= num_steps; i++){
          x = (i-0.5)*step;
          sum = sum + 4.0/(1.0+x*x);
      }
      pi = step * sum;
      printf("PI = %f₩n", pi);
}
```

Compile and Run

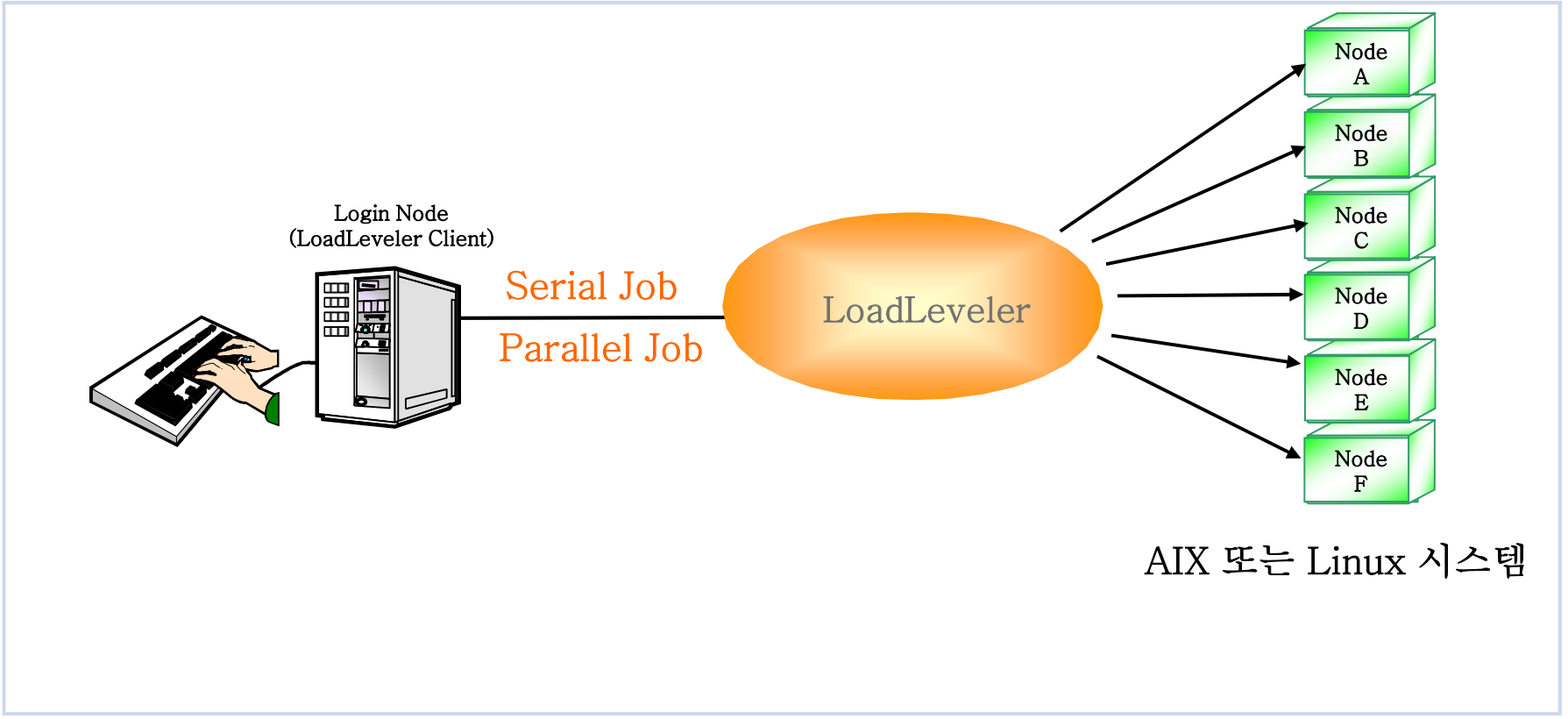
- **C**
  - icc –openmp ppi.c –o ppi
  - export OMP_NUM_THREADS=8
  - time ppi
- **Fortran**
  - ifort –openmp ppi.f90 –o ppi
  - export OMP_NUM_THREADS=8
  - time ppi

LoadLeveler



Loadleveler 개요

- IBM LoadLeveler는 분산 컴퓨터를 위한 일괄 작업(Batch Job) 스케줄러로서 사용자가 제출한 작업을 여러 노드/시스템에 걸쳐 가용한 자원을 자동적으로 할당하여 수행할 수 있게 한다.



Loadleveler 용어

Node	▪ 자체의 운영 체제, CPU 및 메모리를 가지는 독립된 시스템으로 LoadLeveler 클러스터를 구성하는 개별 시스템.
Cluster	▪ 작업 부하 분산을 위해 LoadLeveler가 한 묶음으로 관리하는 노드들의 집합
Class	▪ 제공 자원의 수량 및 특성, 작업 우선 순위, 허용 사용자 등 논리적인 특성을 부여한 가상의 작업 특성
Job Step	▪ 독립적으로 수행되는 작업의 최소 단위. ▪ 각 Job Step은 serial 프로그램일 수도 있고 병렬 프로그램일 수도 있다.

Loadleveler 용어

Job	▪ 한 번에 제출하여 수행하는 Job Step들 전체를 의미하며, 종종 Job Step의 의미로도 쓰인다.
Task	▪ 하나의 작업에 속한 단일 프로세스로서 운영체제가 부여하는 고유한 프로세스 ID를 가지며, 그 자체가 다시 쓰레딩될 수도 있다.
Serial Job	▪ 하나의 노드에서 수행되도록 짜여진 작업. 기본 작업 유형.
Parallel Job	▪ 하나 또는 그 이상의 노드에서 여러 개의 태스크로 나뉘어 수행되도록 짜여진 작업, 즉 다중 노드 및 CPU 작업을 말한다. ▪LoadLeveler는 두 가지의 병렬 작업을 지원한다: ▪PVM: 노드를 할당하고 태스크를 수행하기 위해 PVM API를 지원한다. ▪Parallel: MPI, LAPI 등을 이용하는 분산메모리 병렬 프로그램과 Pthread, OpenMP 등을 이용하는 공유메모리 병렬 프로그램을 지원한다.

Loadleveler를 이용한 작업 제출 및 확인

1. 작업 스크립트 작성: **vi *job_file***
2. 노드 정보 및 상황 출력: **llstatus**
3. 작업 클래스 정보 및 상황 출력: **llclass**
4. 작업 스크립트 제출: **llsubmit *job_file***
5. 큐에 있는 작업의 정보 및 상황 출력: **llq**
6. 큐에서 대기 중인 작업을 중지 상태로 전환: **llhold *job_id***
7. 중지 상태인 작업을 대기 상태 전환: **llhold –r *job_id***
8. 작업 취소: **llcancel *job_id***

Solbaram Loadleveler

```
[p613lsw@solbaram-mg01 ~]$ llclass
Name                 MaxJobCPU     MaxProcCPU   Free   Max Description
                   d+hh:mm:ss     d+hh:mm:ss Slots Slots
-------------- -------------- -------------- ----- ----- --------------------
No_Class             undefined      undefined     0     0                    yes
normal               undefined      undefined   128   136 class for normal parallel jobs yes
--------------------------------------------------------------------------------
"Maximum Slots" value of the class "No_Class" is constrained by the MAX_STARTERS limit(s).
"Free Slots" values of the classes "No_Class", "normal" are constrained by the MAX_STARTERS limit(s).
[p613lsw@solbaram-mg01 ~]$
```

Solbaram Loadleveler status

```
[p613lsw@solbaram-mg01 ~]$ llstatus _R
llstatus: There is currently no machine status to report.
[p613lsw@solbaram-mg01 ~]$ llstatus -R
Machine                        Consumable Resource(Available, Total)
------------------------------ ------------------------------------------------
solbaram-mg01.plsi.or.kr
solbaram01.plsi.or.kr          ConsumableCpus(8,8) ConsumableMemory(24.000 gb,24.000 gb)
solbaram02.plsi.or.kr          ConsumableCpus(8,8) ConsumableMemory(24.000 gb,24.000 gb)
solbaram03.plsi.or.kr          ConsumableCpus(8,8) ConsumableMemory(24.000 gb,24.000 gb)
solbaram04.plsi.or.kr          ConsumableCpus(8,8) ConsumableMemory(24.000 gb,24.000 gb)
solbaram05.plsi.or.kr          ConsumableCpus(8,8) ConsumableMemory(24.000 gb,24.000 gb)
solbaram06.plsi.or.kr          ConsumableCpus(8,8) ConsumableMemory(24.000 gb,24.000 gb)
solbaram07.plsi.or.kr          ConsumableCpus(8,8) ConsumableMemory(24.000 gb,24.000 gb)
solbaram08.plsi.or.kr          ConsumableCpus(8,8) ConsumableMemory(24.000 gb,24.000 gb)
solbaram09.plsi.or.kr          ConsumableCpus(8,8) ConsumableMemory(24.000 gb,24.000 gb)
solbaram10.plsi.or.kr          ConsumableCpus(8,8) ConsumableMemory(24.000 gb,24.000 gb)
solbaram11.plsi.or.kr          ConsumableCpus(0,8) ConsumableMemory(16.188 gb,24.000 gb)
solbaram12.plsi.or.kr          ConsumableCpus(8,8) ConsumableMemory(24.000 gb,24.000 gb)
solbaram13.plsi.or.kr          ConsumableCpus(8,8) ConsumableMemory(24.000 gb,24.000 gb)
solbaram14.plsi.or.kr          ConsumableCpus(8,8) ConsumableMemory(24.000 gb,24.000 gb)
solbaram15.plsi.or.kr          ConsumableCpus(8,8) ConsumableMemory(24.000 gb,24.000 gb)
solbaram16.plsi.or.kr          ConsumableCpus(8,8) ConsumableMemory(24.000 gb,24.000 gb)
solbaram17.plsi.or.kr          ConsumableCpus(8,8) ConsumableMemory(24.000 gb,24.000 gb)
[p613lsw@solbaram-mg01 ~]$
```

Job Script Example: Serial

```
#!/bin/bash
#@ job_type = serial                          ##type is serial
#@ job_name = serialpi                        ##작업 이름 설정
#@ class = normal                             ## 클래스 설정
#@ resources = ConsumableCpus(1)  ConsumableMemory(1000mb)
#@ error = $(job_type)_$(jobid).err           ##에러 파일 이름 설정
#@ output = $(job_type)_$(jobid).out          ##표준 출력 파일 이름 설정
#@ wall_clock_limit = 5:00                    ##예상 작업 소요 시간
#@ queue

time ./spi.out
```

Job Script Example: OpenMP

```
#!/bin/bash
#@ job_type = serial                          ## OpenMP job type is serial
#@ job_name = pi_omp                          ##작업 이름 설정
#@ class = normal                             ##클래스 설정
#@ resources = ConsumableCpus(8)  ConsumableMemory(1000mb)
#@ error = $(job_type)_$(jobid).err           ##에러 파일 이름 설정
#@ output = $(job_type)_$(jobid).out          ##표준 출력 파일 이름 설정
#@ wall_clock_limit = 5:00                    ##예상 작업 소요 시간
#@ queue
export OMP_NUM_THREADS=8
time ./ompi.out
```

Job Script Example: MPI

```
#!/bin/bash
#@ job_type = MPICH                          ## MPI job type
#@ job_name = pi_mpi                         ##작업 이름 설정
#@ class = normal                            ##클래스 설정
#@ resources = ConsumableCpus(1)  ConsumableMemory(1000mb)
#@ error = $(job_type)_$(jobid).err          ##에러 파일 이름 설정
#@ output = $(job_type)_$(jobid).out         ##표준 출력 파일 이름 설정
#@ wall_clock_limit = 5:00                   ##예상 작업 소요 시간
#@ total_tasks = 8
#@ blocking = unlimited
#@ queue
. /etc/profile
module load mpi/intel/openmpi-1.4.3
time mpirun –np $LOADL_TOTAL_TASKS –machinefile $LOADL_HOSTFILE ./mpipi.out
```

Q&A


Thank
you!