

Lab. 7

# Process Management

Duksu Kim



# 공통사항

- 모든 System call 및 Standard C function 사용 가능
  - 단, 외부 라이브러리 사용 불가
- Source code 및 실행파일 이름은 문제 번호 사용
  - 예) 7\_1.c 및 7\_1.out 등
- 시간 측정 필요 시, 자신이 만든 측정 함수 사용
- EL 제출 시,
  - 모든 source code를 모은 한글/워드 파일 별도 제출



Class 01

# Type A

10 조

최선문, 유기대, 황운재



# Lab 7-1. 환경변수 변경 및 추가

## • 문제 설명

- 임의의 환경변수 5개를 입력 받아 기존에 존재하는 환경변수이면 환경 변수의 값을 변경하고 존재하지 않는 환경변수이면 추가한다.
- 업데이트된 환경변수를 파일에 저장한다.  
(추가된 환경 변수와 변경된 환경변수,그 외의 환경 변수 모두 저장)



# Lab 7-1. 환경변수 변경 및 추가

## • 조건

- 입력 받는 5개의 환경변수 중 기존의 존재하는 환경 변수를 적어도 1개 이상 입력할 것
- 입력 받는 5개의 환경변수 중 기존의 존재하지 않는 환경 변수를 적어도 1개 이상 입력하고 환경 변수 이름은 임의로 부여할 것



# Lab 7-1. 환경변수 변경 및 추가

## • 실행 예시

```
root@yugidae:~# ./Lab7_1.out HOME PWD ENK STUDENT SK
환경변수 HOME는 이미 존재하는 환경변수입니다.
환경변수 값을 exist로 변경해주세요.
HOME=exist
환경변수 PWD는 이미 존재하는 환경변수입니다.
환경변수 값을 exist로 변경해주세요.
PWD=exist
환경변수 ENK를 추가합니다.
환경변수 값을 create로 추가해주세요.
ENK=create
환경변수 STUDENT를 추가합니다.
환경변수 값을 create로 추가해주세요.
STUDENT=create
환경변수 SK를 추가합니다.
환경변수 값을 create로 추가해주세요.
SK=create
```

- 기존에 존재하는 환경 변수일 경우  
환경변수 = exist 로 변경
- 기존에 존재하지 않는 환경변수 일 경우  
환경변수 = create 로 추가

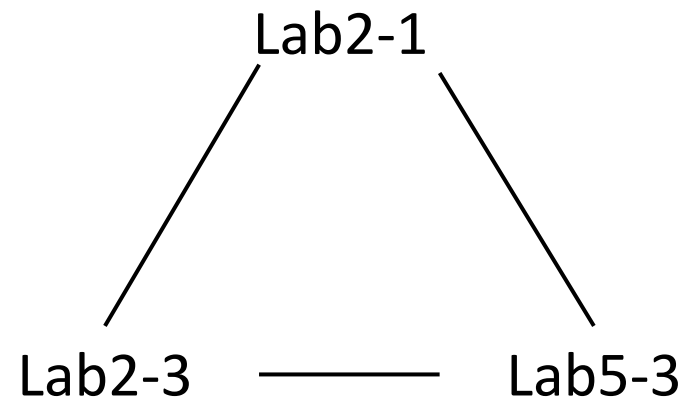




# Lab 7-2. Parallel Programming

## • 문제 설명

교수님께서 남기신 암호를 해독하고,  
학생 계정으로 이를 복사해서 가져오기





# Lab 7-2. Parallel Programming

- 환경설정

```
professor@CSM-PC:~$ ls -al
```

```
$ su professor
```

```
drwx----- 1 professor professor 512 Nov 1 09:56 .secret
```

```
$ mkdir .secret
```

```
$ chmod 4700 .secret
```

- Lab2의 1번과 3번 소스 파일을 가져온 후 컴파일한다.

```
-rw-rw-r-- 1 professor professor 1026 Nov 1 12:00 Copy.c  
-rwxrwxr-x 1 professor professor 9080 Nov 1 12:00 Copy.out  
-rw-rw-rw- 1 root root 826 Nov 1 02:16 Decrypt.c  
-rwxrwxr-x 1 professor professor 8968 Nov 1 09:55 Decrypt.out
```



# Lab 7-2. Parallel Programming

- 실행 예시

```
student@CSM-PC:~$ /home/professor/Lab7_2.out msg.txt
```

```
professor@CSM-PC:~/secret$ ls  
de.txt
```

```
professor@CSM-PC:~/secret$ cat de.txt
```

```
VHPC Lab. is actively recruiting self-motivated M.S. and Ph.D. students.If you are in  
terested in pursuing research on following files as a student member, (1) High Perfor  
mance Computing, GPGPU, Heterogeneous parallel computing, (2) Visualization, Graphics  
, (3) Other interesting research topics (e.g., AI, Block chain, and so on). Please co  
ntact to Prof. Duksu Kim (bluekds (at) koreatech.ac.kr) / https://sites.google.com/vi  
ew/duksukim/recruitprofessor@CSM-PC:~/secret$
```

```
student@CSM-PC:~$ ls
```

```
msg.txt
```

```
student@CSM-PC:~$ cat msg.txt
```

```
VHPC Lab. is actively recruiting self-motivated M.S. and Ph.D. students.If you are in  
terested in pursuing research on following files as a student member, (1) High Perfor  
mance Computing, GPGPU, Heterogeneous parallel computing, (2) Visualization, Graphics  
, (3) Other interesting research topics (e.g., AI, Block chain, and so on). Please co  
ntact to Prof. Duksu Kim (bluekds (at) koreatech.ac.kr) / https://sites.google.com/vi  
ew/duksukim/recruitstudent@CSM-PC:~$
```



# Lab 7-2. Parallel Programming

## • Hints

- 실행 예시와 같게 나오려면 복사해온 Lab2\_1.c 와 Lab2\_3.c를 조금 고쳐야 합니다.
  - 원래 문제에서는 출력이 있었습니다. 출력하는 구문을 삭제하면 됩니다.
- .secret은 소유자만 열 수 있습니다.
- 실행 파일에 setuid를 설정해야 합니다.
- 학생 계정에 쓰려면 EUID를 바꿔야 합니다.
  - EUID, UID => seteuid(), getuid()
- 파일을 읽고 쓰다 보면 경로 문제가 있을 것입니다. 따라서, 절대경로를 사용하세요. 그게 편합니다.



# Lab 7-2. Parallel Programming

- **Hints**

- 각각 만들어진 프로세스(Lab2\_1.out, Lab2\_3.out)은 Lab7\_2.out에서 fork()되어 실행되어야 합니다.
  - execl(), execlp(), execl(), execv(), execvp(), execvpe()
- 각 프로세스의 선후 관계가 있습니다. 즉, 해독이 이루어지고나서, 복사가 이뤄져야 합니다.
  - wait()



# Lab 7-2. Parallel Programming

```
#include <sys/types.h>
#include <sys/wait.h>
```

```
$ man -s 2 wait
```

```
pid_t wait(int* wstatus);
```

- wstatus : 자식 프로세스의 상태
- Return : 종료된 자식 프로세스의 ID 반환, 실패시 -1



```

#include <sys/types.h>
#include <sys/wait.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main(void)
{
    pid_t pid;

    switch (pid = fork())
    {
        case -1:
            // Error
            break;
        case 0:
            // Child
            // Processing...
            // Done
            break;
        default:
            // Parent
            {
                int status;
                pid_t childPid;
                if ((childPid = wait(&status)) < 0)
                {
                    // Error
                }

                // Processing...
                // Done
            }
            break;
    }

    return 0;
}

```



# Lab 7-2. Parallel Programming

- **제출 파일**

- Lab7\_3.c
- Copy.c(Lab2\_1.c)
- Decrypt.c(Lab2\_3.c)

- **make 실행 시 동시에 실행파일 만들어져야 함**



Class 02

# Type B

6조

김동현 김성찬 김영훈 신현우





# Lab 7\_1. 환경 변수 선언

- 7\_1\_1: .bashrc 파일에 환경 변수를 선언하는 프로그램 작성하기
- 7\_1\_2: 7\_1\_1 프로그램을 실행하여 자신의 취미를 환경 변수로 선언하는 프로그램 작성하기



# Lab 7\_1\_1.

## • 문제 설명

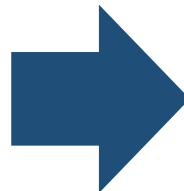
1. 우분투를 새로 실행해도 선언한 환경 변수가 남아 있도록 .bashrc 파일을 수정하는 프로그램 만들기
2. 문자열을 받아서 해당 문자열로 환경변수를 선언  
ex) MY\_HOBBY=SystemProgramming
3. Lab7\_1\_2 프로그램을 통해서 이 프로그램을 실행하도록 할 것



# Lab 7\_1\_1.

## • 실행 결과

```
kdh@DESKTOP-ENG12IL: ~  
# You may want to put all your additional  
# ~/.bash_aliases, instead of adding them  
# See /usr/share/doc/bash-doc/examples/1  
if [ -f ~/.bash_aliases ]; then  
    . ~/.bash_aliases  
fi  
  
# Enable programmable completion features  
# (this must be _before_ 'source' /etc/bash.bashrc)  
if ! shopt -oq posix; then  
    if [ -f /usr/share/bash-completion/bash_completion ]; then  
        . /usr/share/bash-completion/bash_completion  
    elif [ -f /etc/bash_completion ]; then  
        . /etc/bash_completion  
    fi  
fi  
  
export DISPLAY=:0  
export MY_NUMBER=2014136014
```



```
kdh@DESKTOP-ENG12IL: ~  
# source /etc/bash.bashrc  
if ! shopt -oq posix; then  
    if [ -f /usr/share/bash-completion/bash_completion ]; then  
        . /usr/share/bash-completion/bash_completion  
    elif [ -f /etc/bash_completion ]; then  
        . /etc/bash_completion  
    fi  
fi  
  
export DISPLAY=:0  
export MY_NUMBER=2014136014  
  
export MY_HOBBY=SystemProgramming
```



# Lab 7\_1\_2.

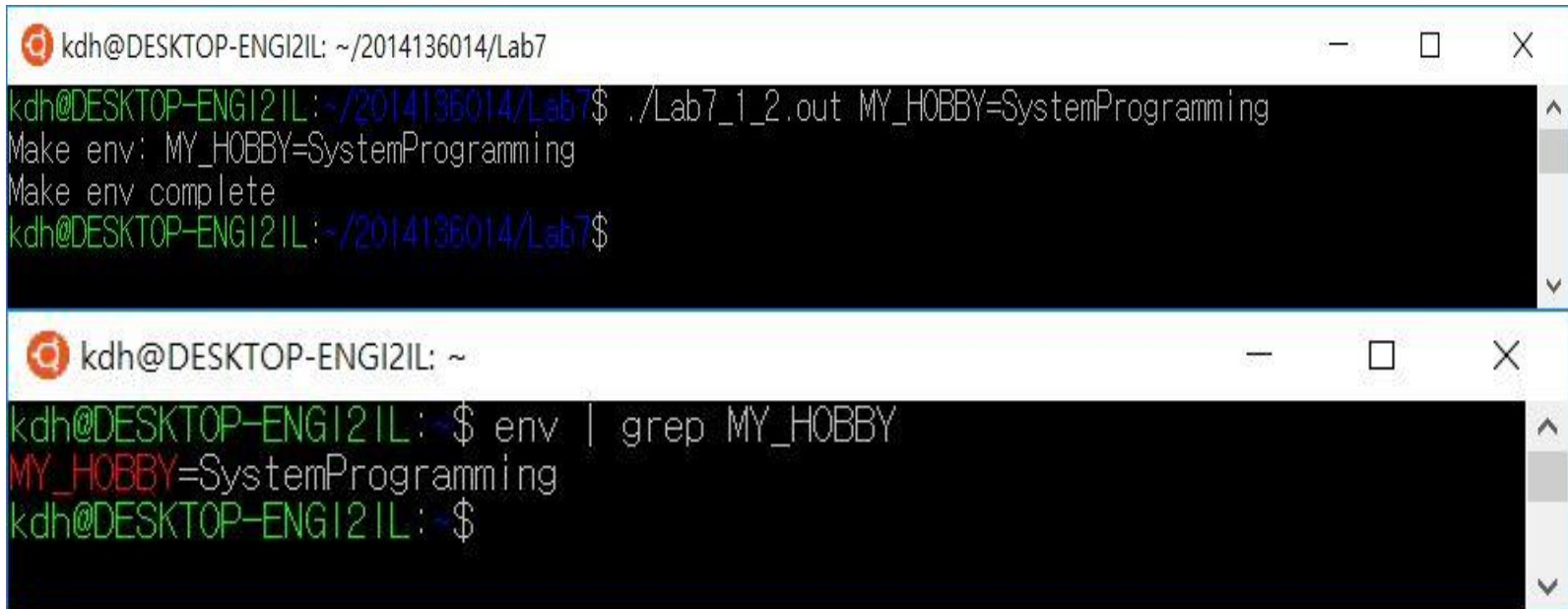
## • 문제 설명

1. 취미를 받아 해당 문자열로 환경 변수를 선언  
ex) `./Lab7_1_2.out MY_HOBBY=SystemProgramming`
2. 프로그램 내부에서 Lab7\_1\_1 프로그램에 적절한 인자를 넘기며 해당 프로그램을 실행
3. 프로그램이 끝나면 우분투를 새로 실행하여 환경 변수가 적절히 선언되었는지 확인



# Lab 7\_1\_2.

- 최종 실행 결과



```
kdh@DESKTOP-ENG12IL: ~/2014136014/Lab7
kdh@DESKTOP-ENG12IL:~/2014136014/Lab7$ ./Lab7_1_2.out MY_HOBBY=SystemProgramming
Make env: MY_HOBBY=SystemProgramming
Make env complete
kdh@DESKTOP-ENG12IL:~/2014136014/Lab7$

kdh@DESKTOP-ENG12IL: ~
kdh@DESKTOP-ENG12IL:~$ env | grep MY_HOBBY
MY_HOBBY=SystemProgramming
kdh@DESKTOP-ENG12IL:~$
```



# Lab 7\_1. 환경변수 선언

- **힌트**

- 경로 설정에 주의할 것
- .bashrc에 환경변수를 추가 후 우분투를 재실행 할 것



# Lab 7\_2. 피보나치 수열

## • 문제 설명

1. fork된 각 프로세스가 피보나치 수열 함수를 실행했을 때 모든 프로세스의 Running time을 출력하는 문제
2. 피보나치 수열의 결과값은 출력하지 않고 PID, PPID, 자식 프로세스의 PID를 출력하고 자식 프로세스가 없다면 none출력
3. Fork횟수와 피보나치 수를 인자로 받음  
- fork횟수  $n=2^n$ 번 포크됨



# Lab 7\_2.

## • 최종 실행 결과

```
kyh9774@DESKTOP-9T1GNFR:~/hw6/Lab7_2$ ./Lab7_2.out 3 9999999
PID      PPID     Child    Running time
22        2        26       0.04s
26        22       none     0.04s
24        22       27       0.05s
27        24       none     0.07s
23        22       28       0.04s
25        23       29       0.05s
29        25       none     0.04s
28        23       none     0.04s
```





# Lab 7\_2.

- **힌트**

- 프로세스가 순서대로 종료되도록 sleep함수 사용
  - 포크 횟수의 절반 만큼 sleep



Common Problem

# Lab 7-3

By DS Kim



# Lab 7-3. Finding Prime Numbers

- 주어진 범위 안에, 소수가 몇 개 있는지 찾는 프로그램 작성
  - 입력 : 수의 범위 (e.g., 1~1,000,000)
- **Serial algorithm**
  - 단일 process로 수행
- **Parallel algorithm**
  - 두개의 process를 이용해서 수행
- **Serial algorithm과 Parallel algorithm의 성능 비교**
- **Hints : fork()**



# Lab 7-3. Finding Prime Numbers

## • 보고서에 포함할 내용

- Serial algorithm과 Parallel algorithm의 성능을 비교
- Parallel algorithm에 대한 고찰을 반드시 포함
  - 미포함 시, 감점 처리

## • 실행의 예

Serial algorithm  
수행 시간

```
$ ./Lab7_3.out 1 1048576
```

```
[Serial start] Sun Nov  4 18:16:41 2018
```

```
[Serial] found 82026 primes
```

```
[Serial end] Sun Nov  4 18:17:42 2018
```

Paralle algorithm  
수행 시간

```
[Parallel start] Sun Nov  4 18:17:42 2018
```

```
[pid = 0] I found 43391 prime numbers between (1 ~ 524287)
```

```
[pid = 0] takes 19222.00 ms
```

```
[Proc.0 end] Sun Nov  4 18:18:02 2018
```

```
[pid = 3092] I found 38635 prime numbers between (524288 ~ 1048576)
```

```
[pid = 3092] takes 51256.21 ms
```

```
[Proc.3092 end] Sun Nov  4 18:18:34 2018
```



Extra Problem

# Lab 7-X

By DS Kim



# 7-X. Efficient Parallel Design

- 소수를 찾는 병렬처리 프로그램 작성
- 입력
  - 검색할 수의 범위
  - 사용할 process의 수
- 출력
  - 각 process 가 찾은 소수의 개수
  - 각 process의 처리 시간
  - 프로그램의 총 소요 시간 계산
    - 수동 or 별도의 프로그램 사용 가능



# 7-X. Efficient Parallel Design

- 입력 받은 process의 수 만큼 process 생성/사용
- 효율적인 병렬처리 알고리즘 설계
  - 프로세스 사이의 부하 불균형 해소
- 목표
  - $Max \left( \frac{t_{program}}{t_{Proc\_i}} \right) < 1.3$ 
    - $t_{Proc\_i}$  : Process i 의 소요 시간
    - $t_{program}$  : Program의 소요 시간
  - 측정 시, 자신의 컴퓨터 물리 core 수 만큼 사용 권장
    - Quad-core CPU 이상 사용 권장 (4개 process 사용)



# 7-X. Efficient Parallel Design

- 7-X 수행 시, 7-1 ~ 7-3 수행 면제
- IPC 사용 불가
- 해설 및 solution code 미제공
- 보고서 작성 시,  
자신이 설계한 병렬처리 알고리즘에 대해 상세  
기술 할 것
- Hints
  - Tree, round-robin

