[실습 8] 쉘과 시작파일

쉘(shell)은 사용자와 운영체제간 인터페이스 역할을 하는 프로그램이다. 최초 쉘인 본 쉘(bourn shell)에서 시작하여 csh (c shell), bash(bourne again shell) 등 쉘 소프트웨어의 종류는 다양하며, 최근에는 zsh이 많은 인기를 끌고 있다. 이번 실습에서는 쉘 전환과 로그인이 일어날 때, 쉘이 실행될 때 자동으로 읽고 실행하는 쉘 스크립트 (shell script) 파일인 시작 파일의 동작을 확인해 본다.

※ 아래 명령어를 이용하여 문제에 답하시오.(1~2)

echo, nano

1) 쉘은 로그인 쉘과 서브쉘로 구분할 수 있다. 로그인 쉘은 telnet이나 ssh로 원격 로그인하거나, 'su - id'로 사용자 전환을 할 때 만들어지는 쉘을 말하고, 서브쉘(subshell)은 데스크 탑 환경에서 터미널 창을 열거나 쉘 프로그램을 실행시키면 만들어지는 쉘이다. 로그인 쉘의 경우 공통 시작 파일인 /etc/profile과 개별 시작 파일인 $HOME/.profile (혹은 $HOME/.bash_profile)을 읽어 실행하고, 서브쉘의 경우 $HOME/.bashrc를 읽어 실행하게 된다. 실습을 통해 각 시작 파일이 언제 어떤 순서로 읽혀지는지 확인해보자. 다음과 같은 결과가 나오도록 각 시작 파일을 적절히 수정 (echo 명령어 라인 삽입) 하시오.

서브쉘	로그인 쉘
~/.bashrc 구동 ubuntu@ubuntu-virtual-machine:~/바탕화면$	/etc/profile 구동 ~/.profile 구동 ~/.bashrc 구동 ubuntu@ubuntu-virtual-machine:~$

2) /etc/profile과 $HOME/.profile의 스크립트를 설명하시오.

[실습 9] 전면 처리와 후면 처리

전면 처리(foreground processing)는 쉘(shell)에서 명령어를 입력하면 명령어 실행이 끝날 때까지 쉘이 기다리며, 실행 중인 명령어는 표준 입력(stdin)을 배타적으로 점유한다. 반면, 후면 처리(background processing)은 명령어를 후면에서 실행함으로써 여러 명령어를 동시에 실행할 수 있게 해 준다. 본 실습을 통해 전면 처리와 후면 처리의 차이를 이해할 수 있다.

※ 아래 명령어를 이용하여 문제에 답하시오.(1~6)

```
cp, mv, source, echo, sleep, jobs, kill
```

☞ firefox로 github 사이트(https://github.com/wkhong89/usys-class)에 접속하여 코드를 다운로드 받으시오.
☞ 다운로드 받은 zip 파일을 같은 디렉토리에 압축해제 하시오.

1) 압축 해제된 파일의 디렉토리 chap05를 로그인 계정의 ~/laPiscine 디렉토리 아래에 복사한 후 디렉터리 이름을 day4로 변경하시오.

☞ day4/bgprocess.sh는 입력 매개변수로 반복 횟수와 반복 주기를 매개값으로 입력받아 반복 횟수를 출력하는 스크립트이다. 예를 들어 1초마다 메시지 출력을 100회 반복하고자 할 경우 사용법은 다음과 같다. (2~6)

```
$ source ./bgprocess.sh 100 1
```

2) 1분 주기 100회 반복과 30초 주기 500회 반복하도록 스크립트를 후면 처리로 실행하시오.

3) 현재 후면 처리로 동작하는 jobs들을 보여주시오.

4) 30초 500회 반복하는 스크립트의 job을 전면 처리로 변경하시오.

5) 30초 500회 반복하는 스크립트의 job을 다시 후면 처리로 변경하시오.

6) 1분 100회 반복하는 스크립트의 job을 강제 종료한 후 종료 결과를 확인하시오.

[실습 10] 입출력 재지정과 파이프

입출력 재지정이란 입력과 출력으로 설정된 기본 파일을 다른 파일로 바꾸는 것을 말하며, 각각 >, <을 이용해 입력과 출력을 다른 파일로 바꿀 수 있다. 파이프는 앞선 명령어의 결과를 뒤에 오는 명령어의 입력으로 전달하는 기능이다. 실습을 통해 사용방법을 익혀본다.

※ 아래 명령어를 이용하여 문제에 답하시오. 실습은 ~/laPiscine/day4에서 진행한다. (1 ~ 8)

```
echo, cat, >, <, >>, |, 2>, ls, wc
```

1) 적절한 명령어를 이용하여 파일 hi.txt와 names.txt의 내용을 합친 새로운 파일 new_file.txt를 생성하는 방법을 2가지 이상 열거하고, 실행결과를 보이시오.

2) 출력 파일에는 표준출력(stdout)과 표준에러(stderr)가 있다. 표준에러는 예외상황에서 발생하는 메시지가 출력되는 파일이다. 표준출력과 표준에러는 모두 같은 화면에 출력되어 구분하기 쉽지 않지만, 표준에러의 경우 예외상황에서 발생하기 때문에 기대하지 않은 출력은 보통 표준에러 메시지라고 보면 된다. ~/laPiscine/day4에서 다음 명령을 실행하여 표준출력과 표준에러 출력을 구분하시오.

```
$ cat hello.c myprg.c
```

3) c언어의 fprintf 함수는 출력 파일을 지정할 수 있다. c 언어에서 모든 파일은 정수형의 descriptor로 식별하는데, 표준입력, 표준출력, 표준에러는 각각 0, 1, 2로 이미 정해져 있다. 다음 프로그램 (파일명: stdout_err.c) 을 작성하고, gcc로 컴파일한 후 실행 결과를 확인해 보시오.

```
#include <stdio.h>
#include <string.h>
#include <unistd.h>

int main(void)
{
    char *std_out = "표준 출력 결과₩n";
    char *std_err = "표준 에러 결과₩n";
    write(1, std_out, strlen(std_out));
    write(2, std_err, strlen(std_err));

    return 0;
}
```

※ gcc 컴파일러가 설치되어 있지 않으면, 다음 명령어를 이용하여 설치하시오.

```
$ sudo apt install gcc
```

※ c 코드 컴파일과 실행 방법

```
$ gcc -o stdout_err stdout_err.c
$ ./stdout_err
```

4) 표준 출력을 파일 out으로 재지정하여 실행한 결과를 보이시오. 표준 출력과 표준 에러 메시지는 각각 어디로 출력되는가?

5) 표준 에러를 파일 err로 재지정하여 실행한 결과를 보이시오. 표준출력과 표준에러 메시지는 각각 어디로 출력되는가?

6) 표준 출력은 파일 out으로 표준 에러는 파일 err로 재지정하는 하나의 명령어를 작성하고 실행결과를 보이시오.

7) 다음 각 명령어의 실행 결과를 설명하시오.

```
$./stdout_err > /dev/null
$./stdout_err 2> /dev/null
$./stdout_err > /dev/null 2>&1
```

8) ls와 wc 명령어, 파이프를 이용하여 ~/laPiscine/day4의 파일의 수를 출력하시오.

[실습 11] 파일 이름과 명령어 대치

명령 프롬프트에서 파일이름을 일일이 열거하지 않고 지정하려는 파일들의 이름에서 공통된 패턴을 찾아 한 번에 여러 파일을 지정할 수 있도록 하는 것을 **파일 이름 대치**라 한다. 와일드 카드 문자 (?, *, [])를 이용하여 파일 이름 대치가 어떻게 이루어지는지 실습을 통해 확인 한다. **명령어 대치**란 문자열에 포함되어 있는 명령어를 그 실행 결과로 대치할 수 있도록 하는 기능이다.

※ 아래 명령어를 이용하여 문제에 답하시오. 실습은 ~/laPiscine/day4에서 진행한다. (1 ~ 5)

echo, ls, mv, cp, wc

1) day4 디렉토리에 있는 file로 시작하는 파일들만 나열하고자 한다. 적절한 명령어를 제시하고 실행결과를 보이시오.

2) day4 디렉토리에 txt_files 디렉토리를 만들고 prac04에 있는 파일 중 txt로 끝나는 파일들만 txt_files로 이동하고자 한다. 적절한 명령어를 제시하고 실행결과를 보이시오.

3) day4 디렉토리에 c_lang 디렉토리를 만들고 hello.c와 std_err.c 파일을 c_lang 디렉토리로 복사하시오. 단, 와일드카드문자를 이용한 파일 이름 대치를 사용하시오.

4) day4 디렉토리에 Files 디렉토리를 만들고 file1과 fil2를 (file33은 제외) files디렉토리로 이동하시오. 단, 와일드카드문자를 이용한 파일 이름 대치를 사용하시오.

5) 결과가 다음과 같이 나오도록 적절한 명령어를 제시하고 실행결과를 보이시오. 단, echo와 wc를 반드시 포함해야 한다.

라인의 수 : 5 names.txt