IO Redirection

작성일시	@2022년 3월 2일 오전 8:30
유형	강의
자료	
복습	☑
강의번호	

IO Redirection

Input Output Redirection

Output

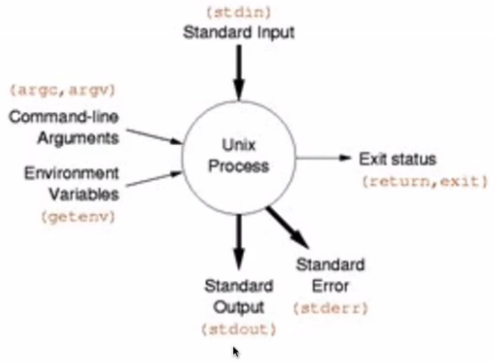
```
ls -l > result.txt
```

현재 ls -l의 결과(터미널에서 보여지는 출력물)을 result.txt로 저장한다.

바로 이것이 우리가 output을, 출력되는 방향을 다른 곳으로 돌려서 파일로 저장을 시킨 것이다. 이것이 바로 redirection 이다.

Review: UNIX Programs

- **Means of input:**
  - Program arguments [control information]
  - Environment variables [state information]
  - Standard input [data]



- **Means of output:**
  - Return status code [control information]
  - Standard out [data]
  - Standard error [error messages]

image

중간에 원으로 표시된게 명령어, 프로그램, 프로세스라고 생각하자. ls, apt-get 등의 프로그램이 동그라미에 해당한다고 생각하면 된다. 프로세스는 크게 입력과 출력을 가지고 있는데 가장 기본적인 입력이 command line argument라고 하는 것이다.

```
ls -al
```

에서 -al 이런 형태의 입력값을 command line argument라고 부른다. 이 프로세스가 실행되면 이 결과를 화면에 출력하게 된다. 이 출력에 해당되는 것이 Standard Output이다. 이것은 모니터에 출력되게 되는데 이걸 redirection 시켜서 다른 곳에 출력시킬 수 있다. 그곳이 바로 파일이다. 또, Standard에도 에러라는게 존재한다. 유닉스 계열의 시스템은 어떤 프로세서가 출력한 결과는 두가지로 구분한다. 하나는 standard output 다른 하나는 오류가 있을때 standard error. 이는 별도의 출력으로 본다.

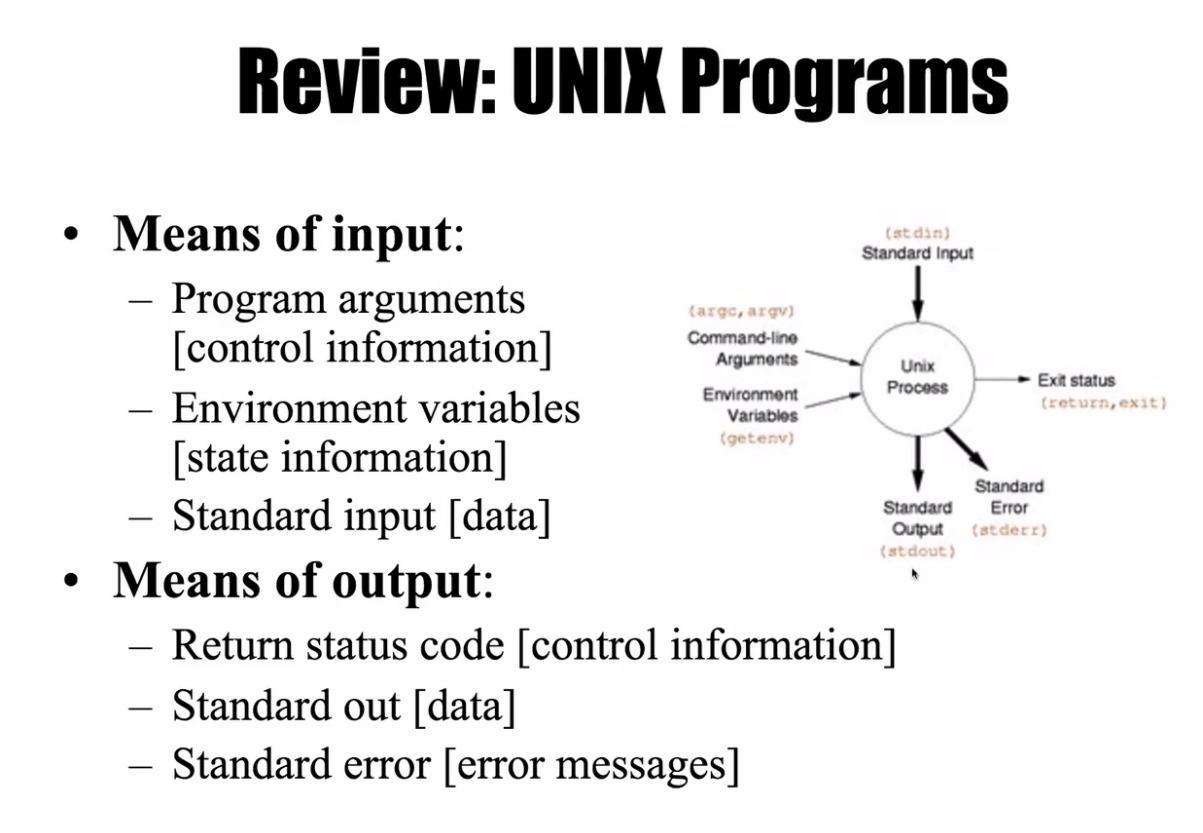
존재하지 않는 파일 temp.txt를 지우려고 하면 error가 발생할 것이다. 이 error 메세지를 redirection 시켜보자.

```
rm temp.txt > result.txt
```

이를 실행하면 우리의 예상과는 다르게 그런 파일이 존재하지 않는 다는 문구가 터미널에 나타나게 된다. 우리가 원했던 것은 문구가 result.txt 파일의 내부에 저장되는 것이었다. 왜 이렇게 된것일까? 왜냐하면 이 문구는 standard output이 아닌 standard error 이기 때문이다! > 는 standard output을 redirection 하겠다는 것이다. standard error을 redirection 하지 않는다.

- 1> : standard output을 redirection
- 2> : standard error을 redirection

Input



image

그림에 보면 한가지의 입력이 표시되어있는데 바로 standard input이다. input 1개와 ouput 2개가 존재하는 것이다.

```
cat hello.txt
```

를 실행하면 cat 프로그램은 result.txt를 두번째 인자로서, 파일명이 들어오면 이 파일의 내용을 보여준다. 그런데, 그냥 cat 만 입력하고 실행하면 프로그램이 끝나지 않고 입력 대기상

태에 있게 된다. 이때 사용자가 입력한 텍스트를 그대로 출력한다.

그러면, standard input이 기본적으로 키보드라면 이 방향을 redirection해서 파일안에 포함되어있는 내용을 cat의 입력값으로 줄 수 있다.

```
cat < hello.txt
```

기본적으로 cat은 키보드의 입력을 받지만 redirection을 시켜주게 되면 hello.txt에 저장된 파일의 내용을 입력으로 받는다.

cat hello.txt와 cat < hello.txt의 차이를 살펴보자. 전자는 cat의 인자를 command line argument 로서 전달한 것이고, 후자는 hello.txt 파일의 내용을 redirection 해서 cat에 전달한 것이다.

standard input의 다른 예시도 살펴보자.

아주 많은 텍스트를 가진 파일 linux.txt가 존재한다.

```
head linux.txt
```

를 실행하면 linux.txt의 모든 내용을 출력하지 않고 기본 10줄만 출력해준다.

```
head -n1 linux.txt
```

를 실행하면 한 줄만 출력해준다. 이때, -n1는 command line argument로서 head에 전달한 인자다. 그럼 standard input redirection의 방법으로는 어떻게 해야할까?

```
head -n1 < linux.txt
```

로 하면 된다! 이 결과물을 standard output redirection으로 파일에 저장하고 싶다면?

```
head -n1 < linux.txt > one.txt
```

정리를 해보면, 유닉스 계역의 시스템에서 프로세스는 standard input, standard output, standard error 의 흐름을 IO Stream이라고 부른다.

append

```
ls -al > result.txt
```

를 실행하면 현재의 ls -al의 아웃풋이 result.txt에 저장된다. 다시 이 문장을 실행하면 어떻게 될까? 추가된 result.txt 파일까지의 결과가 다시 result.txt에 덮어쓰게 된다.

```
ls -al >> result.txt
```

덮어쓰기가 아니라 새로 생긴 파일이 뒤에 추가되는 append 되게 하고 싶다면 >>를 사용하면 된다.

그럼 << 도 확인해보자.

```
mail hyeonkang.dev@gmail.com << eot
> hi
> nice
> to
> see
> you
> eot
```

이렇게 입력하면 내가 입력한 여러개의 정보들이 mail이라는 프로세스의 인풋으로 redirection 되는 것이다. << 뒤에 오는 문자의 의미는 이 문자가 입력되면 거기서 입력이 종료된다는 것이다를 의미하는 특수한 기호이다. 문자가 eot가 아니어도 상관이 없다.

출력되는 결과가 터미널에도 나오지않고 파일로도 저장되지 않게 하고 싶을때는 어떻게 해야 할까?

```
ls -al > /dev/null
```

/dev/null 은 유닉스 계열에서는 쓰레기통 같은 것이다. 이 실행결과가 /dev/null 로 redirection 되면 그 결과물이 그냥 낭떠러지로 떨어진다고 생각하면 된다.