실험 UNIX-1: 예비보고서

전공: 철학과 학년: 3학년 학번: 20180085 이름 : 이길원

1. 목적

유닉스 시스템에 대하여 미리 접해본 후 실험에 임할 수 있도록 한다. 아울러 부록에 나와 있는 명령어에 대하여 익숙해지도록 사용해본다.

2. 예비 학습

UNIX 시스템에 접속해본 뒤 자신의 홈 디렉토리를 확인해본다.

```
홈 디렉토리 : /sogang/under/cse20180085
```

쉘 프로그래밍 실험에서 사용할 데이터 파일인 전화번호부를 만들어본다. 단 데이터 파일의 형식은 실험에서 나온 예제에 따르도록 한다. 5명 이상이 들어가 있는 데이터를 만들되 vi 에디터를 이용하여 작성한다. 단 파일명은 data로 한다.

```
(데이터 파일)

홍길동|서울시 마포구 신수동 서강대학교 AS관 301호|02-705-2665
Andres|경기도 의정부시 호원동 23-12번지|031-827-7942
Draw|서울시 마포구 신수동 서강대학교 R관 914호|010-123-4567
이길원|서울시 마포구 광성로 24길|010-2397-3829
조석제|충청북도 청주시 서원구 창직로 23|010-3113-3245
```

위의 예제를 편집하는데 사용한 vi 명령어들을 나열하고, 해당 명령 수행하는 결과를 적어보도록 한다.

```
shell : vi data  -> vi 에디터를 실행하여 data 이름을 가진 파일로 이동한다.

vi : '-i' -> INSERT 모드로 변경

vi : ':wq' -> 파일을 저장 후 에디터를 종료
```

위에서 작성한 데이터 파일을 $home/.data 파일로 복사한다. 복사하기 위하여 사용한 명령들을 적어보도록 한다.

```
cse20180085@cspro:~$ cp data $HOME/.data
```

$home/.data 파일을 그룹 및 다른 사용자가 아무 권한도 갖지 않도록 권한 변경을 해본다. 사용한 명령을 적어보도록 한다.

```
ccse20180085@cspro:~$ chmod 700 $HOME/.data

-> User만 읽기(4), 쓰기(2), 실행(1) 권한을 가지며 이외의 사용자 또는 그룹은 .data 파일에 대해서 아무 권한도 갖지 못한다.
```

디렉토리에 대한 읽기, 쓰기, 실행 권한을 설정해보고 각각이 갖는 의미를 살펴본다.

chmod 400 temp -> temp 디렉토리 내부의 파일, 하위 디렉토리를 읽을 수 있음, 그러나 읽기(-r)만 가능하고, 실행(-x) 가능하지 않다면 temp디렉토리에 접근 할 수 없음. 또한

chmod 200 temp -> temp 디렉토리에 새로운 파일을 생성하거나 기존 파일, 디렉토리를 수정할 수 있음. 그러나 쓰기(-w)만 가능하다면, 실행(-x)가능하지 않아 temp에 접근할 수 없고, 읽기(-r) 또한 할 수 없음.

chmod 100 temp -> temp 디렉토리에 접근을 가능하게 해줌. 그러나, 실행(-x)만 가능하다면, 읽기(-r) 권한이 없어 디렉토리를 구성하는 파일 또는 하위 디렉토리를 볼 수 없으며, 쓰기 (-w) 권한이 없어 새로운 파일을 생성하거나 수정할 수 없음.

3자리 수를 통해 사용자, 권한을 지정할 수 있다. 첫 번째 자리는 user(현재 사용자), 두 번째 자리는 group(그룹 멤버), 세 번째 자리는 other(user, group 이외 사용자) 에 대한 권한을 지정한다. 다음으로 4는 read 권한, 2는 write 권한, 1은 execute 권한을 부여한다. 각 권한에 부여된 수를 bitwise OR 연산 하여 모든 권한의 조합을 표현할 수 있다.

3. 보충 학습

Regular Expression에 대하여 정리해보도록 한다.

Regular Expression은 특정한 규칙을 가진 문자열의 집합을 표현하는 데 사용되는 형식 언어이다. 실질적으로 Regular Expression 은 다양한 문서 편집기, 워드 프로세서 또는 검색 엔진 등에서 문자열 검색, 치환을 위해 사용된다.

구체적으로 Regular Expression은 “패턴”을 정해 이를 준수하는 문자열들을 찾는다. 이 때 “패턴”을 정하기 위해 크게 메타문자, 정규문자가 사용된다. 메타 문자는 특별한 의미를 갖는 문자이며, 정규문자는 리터럴, 즉 문자 그대로의 의미를 갖는 것으로 해석된다. 메타 문자에는 가장 널리 쓰이는 ‘*’(와일드카드) 부터, ‘.’, ‘^’, ‘$’, ‘[]’ 등 다양한 문자들이 존재한다. 와일드카드는 일반적으로 0개 이상의 어떤 리터럴과 대응되는 경우를 표현한다. 또한 ₩b, ₩d, ₩s , ₩B, ₩D, ₩S 등의 메타 문자도 존재한다. ₩b는 공백, 탭, 컴마 등 단어를 구분해주는 문자와 매칭된다. ₩B는 이 외의 경우들과 매칭된다. ₩d는 숫자와 매칭되며 [0-9]와 같은 표현이다. ₩D는 ₩d와 매칭되는 경우를 제외한 모든 경우에 매칭된다. 이러한 메타 문자와 정규 문자를 조합하여 우리는 “패턴”을 만들 수 있고, 이 “패턴”을 통해 원하는 문자열들의 집합을 표현할 수 있다.

Regular Expression의 강점은 위에서 설명한 메타 문자에서 확인할 수 있다. 예를 들어 우리가 한글 파일을 찾고 싶은 경우를 생각해 볼 수 있다. 이 때, 우리는 한글 파일의 확장자 명이 .hwp임을 알고 있다. 이런 경우 간단히 “*.txt” 라는 패턴을 통해 모든 한글 파일명들의 집합을 표현할 수 있다.

사실 이런 Regular Expression 는 정해진 기준 없이 중구난방으로 사용하는 경우가 많았다. 하지만 최근엔 POSIX 기본 및 POSIX 확장 표준 문법을 기준으로 활용되는 경우가 많다.