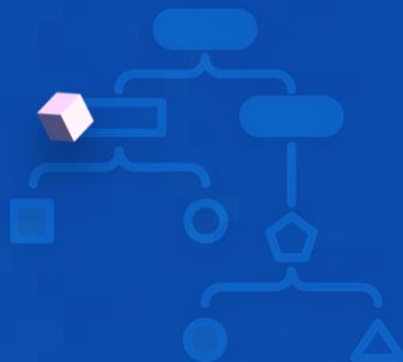


강원혁신플랫폼

리눅스프로그래밍

셸 프로그래밍





bash 셸에서 프로그램의 반복구조에 사용되는 키워드가 아닌 것은 무엇인가?

- ☒ 1 do while ☐ 2 while ☐ 3 for ☐ 4 until

1

bash 셸의 프로그램 반복구조 처리에는 while, for, until이 사용된다.



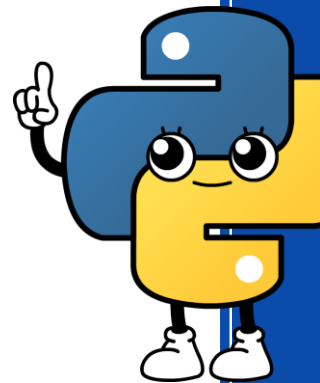


학습 내용

- 1 셀 프로그래밍 수식표현
- 2 셀 프로그래밍 제어 구조 종류
- 3 셀 프로그래밍 반복 구조 종류
- 4 셀 프로그래밍 흐름제어 구조 종류

학습 목표

- 📖 셀 프로그래밍 수식표현에 대해 설명할 수 있다.
- 📖 셀 프로그래밍 제어, 반복, 흐름제어 구조에 대해 설명할 수 있다.



강원혁신플랫폼

리눅스프로그래밍



셸 프로그래밍 수식표현





셸 프로그래밍

[셸 스크립트를 작성하여 리눅스 시스템을 자동화하고
작업을 자동화하는 프로세스
일련의 셸 명령어와 제어 구조를 포함하는 스크립트]



파일 및 디렉토리 관리



- ◆ 파일 및 디렉토리를 생성, 삭제, 이동, 복사 등을 자동화



셸 프로그래밍

01

02

03

04

05

06

07

프로세스 관리



- ◆ 백그라운드에서 실행되는 프로세스를 시작, 중지 또는 모니터링



01

02

03

04

05

06

07

시스템 설정 및 관리



- ◆ 시스템 설정을 변경하거나 관리 작업을 자동화



셸 프로그래밍

01

02

03

04

05

06

07

데이터 처리



- ◆ 셸 스크립트를 사용하여 텍스트 파일의 내용을 처리하거나 필터링



01

02

03

04

05

06

07

조건문 및 반복문 사용



- ◆ 조건문(if-else) 및 반복문(for, while)을 사용하여 특정 조건에 따라 프로그램의 흐름을 제어하거나 반복 작업 수행



웹 프로그래밍

01

02

03

04

05

06

07

함수 정의



- ◆ 함수를 정의하여 관련 코드 블록을 그룹화하고 재사용



셸 프로그래밍

01

02

03

04

05

06


07

명령어 및 변수 활용




- ◆ 다양한 셸 명령어를 활용하여 작업을 수행하고, 변수를 사용하여 데이터를 저장하고 전달

“Hello World!” 셸 프로그래밍

 셸 스크립트 편집, nano 에디터 사용 가능

```
$ vi helloworld.sh
#!/bin/sh
echo "Hello World\n"
```

“Hello World!” 셸 프로그래밍


 작성된 파일 확인

```
$ ls  
helloworld.sh
```

스크립트 실행 불가

```
$ ./helloworld.sh  
-bash: ./helloworld.sh:  
Permission denied
```

“Hello World!” 셸 프로그래밍

 셸 이용 스크립트 실행 가능

```
$ sh helloworld.sh  
Hello World
```


“Hello World!” 셸 프로그래밍

⊕ 셸 이용 스크립트 경로지정 실행 가능

```
$ sh ./helloworld.sh  
Hello World
```

“Hello World!” 셸 프로그래밍

 셸 스크립트를 실행파일로 변경

```
$ chmod +x helloworld.sh
```

경로지정 직접 실행


```
$ ./helloworld.sh  
Hello World
```

“Hello World!” 셸 프로그래밍

⊕ echo를 printf로 변경 가능

```
$ vi helloworld.sh
#!/bin/sh
printf " Hello World\n "
$ ./helloworld.sh
Hello World
```

셀 프로그래밍 수식표현

 수식은 산술 연산, 비교 연산, 논리 연산 등을 포함

산술 연산

덧셈 • $a = \$((3 + 4))$

뺄셈 • $b = \$((10 - 5))$

곱셈 • $c = \$((2 * 6))$

나눗셈 • $d = \$((20 / 5))$

나머지 • $e = \$((10 \% 3))$

변수 사용

변수와 함께 연산 • $result = \$((a + b))$

변수에 할당된 값을 활용 • $a = 5; b = \$((a * 2))$



셸 프로그래밍 수식표현

수식은 산술 연산, 비교 연산, 논리 연산 등을 포함

비교 연산

같음

• if [\$a -eq \$b]; then ... fi

큼

• if [\$a -gt \$b]; then ... fi

같지 않음

• if [\$a -ne \$b]; then ... fi

작거나 같음

• if [\$a -le \$b]; then ... fi

작음

• if [\$a -lt \$b]; then ... fi

크거나 같음

• if [\$a -ge \$b]; then ... fi



셸 프로그래밍 수식표현

수식은 산술 연산, 비교 연산, 논리 연산 등을 포함

논리 연산

AND • if [\$a -eq 5] && [\$b -gt 10]; then ... fi

OR • if [\$a -eq 5] || [\$b -gt 10]; then ... fi

NOT • if ![\$a -eq 5]; then ... fi

변수값을 참조할 때 \$를 사용

강원혁신플랫폼

리눅스프로그래밍



셸 프로그래밍 제어 구조 종류





웹 프로그래밍 제어 구조 if

📦 웹 프로그래밍

- ♦ if 문은 조건에 따라 특정 작업을 수행하는 데 사용

참(True)인 경우

특정 코드 블록을 실행

거짓(False)인 경우

다른 코드 블록을 실행하거나 넘어감



웹 프로그래밍 제어 구조 if

⊕ 일반적인 if 문의 구문

```
if [ 조건 ]; then  
    # 조건이 참일 때 수행할 코드  
else  
    # 조건이 거짓일 때 수행할 코드  
fi
```



셸 프로그래밍 제어 구조 if



비교 연산자



= (같음), != (같지 않음), -eq (같음), -ne (같지 않음), -lt (작음),
-gt (큼), -le (작거나 같음), -ge (크거나 같음) 등



셸 프로그래밍 제어 구조 if

01

02

03

04

문자열 비교



= (같음), != (같지 않음)



셸 프로그래밍 제어 구조 if

01

02

03

04

파일 및 디렉토리 존재 확인



-f (파일 존재 여부), -d (디렉토리 존재 여부)



셸 프로그래밍 제어 구조 if

01

02

03

04

논리 연산자



-a (AND), -o (OR), ! (NOT)



셸 프로그래밍 제어 구조 if

⊕ 간단한 if 문의 예시

```
if [ $age -ge 18 ]; then  
    echo "성인입니다."  
else  
    echo "미성년자입니다."  
fi
```



셸 프로그래밍 제어 구조 if

조건에 따라 if-elif-else 문 사용

```
if [ $score -ge 90 ]; then
    echo "A 학점입니다."
elif [ $score -ge 80 ]; then
    echo "B 학점입니다."
elif [ $score -ge 70 ]; then
    echo "C 학점입니다."
else
    echo "D 학점입니다."
fi
```

\$score 변수의 값에 따라 학점을 출력

조건문은 셸 프로그래밍에서 매우 유용하며, 복잡한 조건을 처리하거나 다양한 분기를 제어하는 데 사용

셸 프로그래밍 제어 구조 case

셸 프로그래밍 case

- ◆ case 문은 다양한 경우(case)에 따라 코드 블록을 실행하는 데 사용
- ◆ case 문은 주로 특정 변수 또는 값에 대한 다중 분기를 처리하는데 유용
- ◆ 각 경우에 대한 패턴을 지정하고 해당 패턴과 일치하는 경우에 특정 코드 블록을 실행

일반적인 case 문의 구문

```
case 변수 in
  패턴1)
    # 패턴1과 일치하는 경우 실행할 코드
    ;;
  패턴2)
    # 패턴2와 일치하는 경우 실행할 코드
    ;;
  패턴3)
    # 패턴3과 일치하는 경우 실행할 코드
    ;;
  *)
    # 모든 경우에 일치하지 않는 경우 실행할 코드
    ;;
esac
```

case 문 예시

```
fruit="apple"
```

```
case $fruit in
```

```
  "apple")
```

```
    echo "사과입니다."
```

```
    ;;
```

```
  "banana")
```

```
    echo "바나나입니다."
```

```
    ;;
```

```
  "orange")
```

```
    echo "오렌지입니다."
```

```
    ;;
```

```
  *)
```

```
    echo "알 수 없는 과일입니다."
```

```
    ;;
```

```
esac
```

강원혁신플랫폼

리눅스프로그래밍



셸 프로그래밍 반복 구조 종류





웹 프로그래밍 반복 구조 for

웹 프로그래밍 for

- ◆ for 문은 주어진 목록 또는 범위에 대해 반복 작업을 수행하는 데 사용
- ◆ for 문은 목록의 각 항목에 대해 지정된 코드 블록을 실행



웹 프로그래밍 반복 구조 for

⊕ 일반적인 for 문의 구문

```
for 변수 in 값1 값2 값3 ...; do  
    # 변수를 사용하여 실행할 코드  
done
```



셸 프로그래밍 반복 구조 for

for 문 예시

```
for fruit in apple banana orange; do  
    echo "과일: $fruit"  
done
```

"apple", "banana", "orange"를 각각 \$fruit 변수에 할당하고, echo를 사용하여 각 과일에 대한 메시지를 출력



셸 프로그래밍 반복 구조 for

숫자 범위를 사용하는 예시

```
for num in {1..5}; do  
  echo "숫자: $num"  
done
```

1부터 5까지의 숫자 범위를 반복하고, 각 숫자를 \$num 변수에 할당하여 출력

- ◆ for 문은 파일 리스트, 디렉토리 목록, 명령 결과 또는 정수 범위와 같은 다양한 항목에 대해 반복 작업을 수행하는 데 사용
- ◆ 반복적인 작업을 자동화하거나 목록의 각 항목에 대한 동일한 작업을 수행

셸 프로그래밍 반복 구조 while

셸 프로그래밍 while

- ◆ while 문은 주어진 조건이 참(True)인 동안 코드 블록을 반복해서 실행
- ◆ while 문은 조건을 평가하고, 조건이 참일 경우 코드 블록을 실행하고 다시 조건을 평가하는 과정을 반복

일반적인 while 문의 구문

```
while 조건; do  
    # 조건이 참일 때 실행할 코드  
done
```

조건은 참 또는 거짓을 평가
코드 블록은 조건이 참일 때 실행되며, 반복이 필요한 동안 조건이 참인지 확인

while 문 예시

```
counter=1
```

```
while [ $counter -le 5 ]; do  
    echo "카운터: $counter"  
    counter=$((counter + 1))  
done
```

- ◆ \$counter 변수가 1부터 5까지 증가하는 동안 코드 블록을 반복해서 실행
- ◆ echo를 사용하여 각 카운터 값을 출력하고, counter 변수를 증가
- ◆ while 문은 조건이 참인 동안 반복 작업을 수행해야 할 때 유용
- ◆ while 문을 사용하여 데이터 처리, 파일 읽기, 무한 루프 등을 구현
- ◆ 반복 작업을 제어하고 조건을 업데이트하여 필요에 따라 반복을 중지

셸 프로그래밍 반복 구조 until

셸 프로그래밍 until

- ◆ until 문은 주어진 조건이 거짓(False)인 동안 코드 블록을 반복해서 실행하는 데 사용
- ◆ until 문은 조건을 평가하고, 조건이 거짓일 경우 코드 블록을 실행하고 다시 조건을 평가하는 과정을 반복

일반적인 until 문의 구문

```
until 조건; do  
    # 조건이 거짓일 때 실행할 코드  
done
```

- ◆ 조건은 참 또는 거짓을 평가
- ◆ 코드 블록은 조건이 거짓일 때 실행되며, 반복이 필요한 동안 조건이 거짓으로 평가되는지 확인

until 문의 예시

```
counter=1
```

```
until [ $counter -gt 5 ]; do  
    echo "카운터: $counter"  
    counter=$((counter + 1))  
done
```

- ◆ \$counter 변수가 5보다 큰 동안 코드 블록을 반복해서 실행
- ◆ echo를 사용하여 각 카운터 값을 출력하고, counter 변수를 증가
- ◆ until 문은 조건이 거짓인 동안 반복 작업을 수행해야 할 때 유용
- ◆ until 문을 사용하여 데이터 처리, 파일 읽기, 무한 루프 등을 구현
- ◆ 반복 작업을 제어하고 조건을 업데이트하여 필요에 따라 반복을 중지

강원혁신플랫폼

리눅스프로그래밍



셸 프로그래밍 흐름제어 구조 종류



셸 프로그래밍 흐름제어 break

셸 프로그래밍 break

- ◆ break 문은 반복문에서 루프를 강제로 종료하는 데 사용
- ◆ 특정 조건이 충족되었을 때 반복문을 종료
- ◆ break 문은 if 문과 함께 사용되어 특정 조건이 만족될 때 반복문을 종료

④ 1부터 10까지 숫자를 출력하면서 숫자 5를 만났을 때 반복문을 종료

```
for ((i=1; i<=10; i++)); do
    echo $i
    if [ $i -eq 5 ]; then
        break
    fi
done
```

break 문은 for, while, until과 같은 반복문에서 사용 가능

셸 프로그래밍 흐름제어 continue

셸 프로그래밍 continue

- ◆ continue 문은 반복문에서 현재 반복을 중단하고 다음 반복을 시작하는 데 사용
- ◆ continue 문을 실행하면 현재 반복의 남은 코드를 건너뛰고 다음 반복으로 진행
- ◆ continue 문은 if 문과 함께 사용되어 특정 조건이 충족되었을 때 남은 코드를 건너뛰고 다음 반복으로 이동

⊕ 1부터 5까지의 숫자 중 홀수일 때만 출력하고 짝수일 때는 수행을 생략

```
for ((i=1; i<=5; i++)); do
  if ((i % 2 == 0)); then
    continue
  fi
  echo $i
done
```

continue 문은 for, while, until과 같은 반복문에서 사용 가능



- 셸 프로그래밍

- ◆ 셸 프로그래밍 수식표현
- ◆ 셸 프로그래밍 제어 구조 종류
- ◆ 셸 프로그래밍 반복 구조 종류
- ◆ 셸 프로그래밍 흐름제어 구조 종류