

دانشگاه صنعتي امیرکبیر  
(پلی تکنیک تهران)

دانشكده مهندسی برق

آزمایشگاه سیستم عامل

آزمایش ۸

شبیه ساز الگوریتم های زمان بندی

نگارش

علی بابالو

پویا شریفی

استاد راهنما

مهندس کیخا

خرداد ماه 1402

در کد Bash، `$0` به نام اسکریپت یا برنامه‌ای که در حال اجرا است اشاره دارد. این نمایانگر نام دستور یا نام اسکریپت خود است.

هنگام استفاده از `$0` در یک اسکریپت Bash، آن با نام فایل اسکریپت جایگزین می‌شود. به عنوان مثال، اگر اسکریپت شما `myscript.sh` نام داشته باشد، `$0` به `myscript.sh` تبدیل می‌شود.

به طریق مشابه، اعداد دیگر و پرانتزهای زوج برای اهداف مختلفی در کد Bash مورد استفاده قرار می‌گیرند:

- `$1`، `$2`، `$3` و غیره: این متغیرها نمایانگر آرگومان‌های خط فرمانی که به اسکریپت یا تابع پاس داده شده‌اند هستند. `$1` نمایانگر اولین آرگومان، `$2` نمایانگر دومین آرگومان و به همین ترتیب عمل می‌کنند.

- `${}`: پرانتزهای زوج در کد Bash برای توسعه پارامتر یا جایگزینی متغیرها استفاده می‌شوند. این پرانتزها به شما امکان می‌دهند عملیات‌های مختلفی روی متغیرها انجام دهید، مانند دسترسی به مقادیر آنها، انجام عملیات رشته‌ای یا استفاده از مقادیر پیش‌فرض در صورتی که متغیر تنظیم نشده باشد.

به عنوان مثال، `${var}` مقدار متغیر `var` را نمایش می‌دهد، `${#var}` طول مقدار ذخیره شده در `var` را برمی‌گرداند و `${var:-default}` از مقدار `var` استفاده می‌کند اگر مقدار آن تنظیم شده باشد و در غیر این صورت از مقدار پیش‌فرض `default` استفاده می‌کند.

این نمادها و نحوه نوشتار ویژه، بخشی از قابلیت‌های قدرتمند Bash هستند و امکان کنترل قدرتمندی بر روی متغیرها و رفتار اسکریپت را فراهم می‌کنند.

در زمینهٔ گزینه‌ها یا آرگومان‌های خط فرمان، معنای `-s` و `-sp` بسته به دستور یا برنامهٔ خاصی که استفاده می‌شود، متفاوت است. با این حال، می‌توانم برخی تفسیرهای رایج برای این گزینه‌ها را بیان کنم:

1. `-s`: این گزینه معمولاً برای تعیین منبع یا ورودی در یک دستور یا برنامه استفاده می‌شود. می‌تواند به معنای "منبع" یا "منبع ورودی" باشد. به عنوان مثال:

- در یک دستور مرتبط با مدیریت فایل، `-s` می‌تواند فایل یا پوشهٔ منبع را مشخص کند.

- در یک دستور شبکه‌سازی، `-s` می‌تواند آدرس IP یا نام میزبان منبع را مشخص کند.

- در یک دستور پردازش داده، `-s` می‌تواند داده یا جریان ورودی را مشخص کند.

2. `-sp`: این گزینه یک گزینهٔ استاندارد رایج نیست و تفسیر آن ممکن است به وابستگی به زمینه یا برنامهٔ خاص، متفاوت باشد. بدون اطلاعات بیشتر، منتشر کردن یک معنای خاص برای `-sp` دشوار است. احتمالاً `-sp` می‌تواند یک گزینهٔ سفارشی یا مرتبط با برنامه‌ای باشد که در یک دستور یا اسکریپت خاص استفاده می‌شود.

برای دریافت اطلاعات دقیق و جزئی در مورد معنای این گزینه‌ها در یک دستور یا برنامهٔ خاص، توصیه می‌شود به مستندات و منابع کمکی مربوط به آن دستور یا برنامه مراجعه کنید.

خروجی کد:

/Desktop/3$ ./1.sh

18

20

1

1

7

10

11

سوال اول:

خروجی کد 2:

#!/bin/bash

# Check if two arguments are provided

if [ $# -ne 2 ]; then

  echo "Error: Two arguments are required."

  exit 1

fi

# Check if the arguments are valid numbers

re='^[0-9]+$'

if ! [[ $1 =~ $re ]] || ! [[ $2 =~ $re ]]; then

  echo "Error: Invalid input. Arguments must be numeric."

  exit 1

fi

# Convert the arguments to integers

num1=$1

num2=$2

# Print the larger number

if [ $num1 -gt $num2 ]; then

  echo "Larger number: $num1"

else

  echo "Larger number: $num2"

fi

# Add the numbers

sum=$((num1 + num2))

echo "Sum: $sum"

pooya@DESKTOP-741TI2M:/mnt/c/Users/shadi/Desktop/3$ ./2.sh 10 20

The larger number is: 20

The sum is: 30

pooya@DESKTOP-741TI2M:/mnt/c/Users/shadi/Desktop/3$ ./2.sh 20 30

Larger number: 30

Sum: 50

pooya@DESKTOP-741TI2M:/mnt/c/Users/shadi/Desktop/3$ ./2.sh 20 @

Error: Invalid input. Arguments must be numeric

سوال دوم:

#!/bin/bash

# Function to perform addition

add() {

  sum=$(($1 + $2))

  echo "Result: $sum"

}

# Function to perform subtraction

subtract() {

  diff=$(($1 - $2))

  echo "Result: $diff"

}

# Function to perform multiplication

multiply() {

  product=$(($1 \* $2))

  echo "Result: $product"

}

# Function to perform division

divide() {

  if [ $2 -eq 0 ]; then

    echo "Error: Division by zero is not allowed."

  else

    quotient=$(($1 / $2))

    echo "Result: $quotient"

  fi

}

# Read user input

read -p "Enter the operation (+, -, \*, /): " operation

read -p "Enter the first number: " num1

read -p "Enter the second number: " num2

# Perform the selected operation

case $operation in

  "+")

    add $num1 $num2

    ;;

  "-")

    subtract $num1 $num2

    ;;

  "\*")

    multiply $num1 $num2

    ;;

  "/")

    divide $num1 $num2

    ;;

  \*)

    echo "Error: Invalid operation."

    ;;

esac

برای هر یک از دستورات یک فانکشن زدیم و سپس تک به تک ورودی ها را گرفتین و با استفاده از case دستور مشخص شده را روی اعداد ورودی انجام دادیم.

خروجی:

pooya@DESKTOP-741TI2M:/mnt/c/Users/shadi/Desktop/3$ ./3.sh

Enter the operation (+, -, \*, /): +

Enter the first number: 20

Enter the second number: 234

Result: 254

pooya@DESKTOP-741TI2M:/mnt/c/Users/shadi/Desktop/3$ ./3.sh

Enter the operation (+, -, \*, /): \*

Enter the first number: 20

Enter the second number: 6

Result: 120

سوال3:

#!/bin/bash

# Function to reverse a number

reverse\_number() {

  number=$1

  reversed=""

  while [ $number -gt 0 ]; do

    remainder=$((number % 10))

    reversed="${reversed}${remainder}"

    number=$((number / 10))

  done

  echo "Reversed number: $reversed"

}

# Read user input

read -p "Enter a number: " num

# Call the function to reverse the number

reverse\_number $num

./4.sh

Enter a number: 6547

Reversed number: 7456

سوال 4:

#!/bin/bash

# Function to read lines from a file between two line numbers

read\_lines() {

  start\_line=$1

  end\_line=$2

  file=$3

  # Validate file existence

  if [ ! -f "$file" ]; then

    echo "Error: File '$file' does not exist."

    exit 1

  fi

  # Validate line numbers

  if [ $start\_line -gt $end\_line ]; then

    echo "Error: Invalid line numbers."

    exit 1

  fi

  # Read and print lines from the file

  awk "NR >= $start\_line && NR <= $end\_line" "$file"

}

# Read user input

read -p "Enter the start line number: " start

read -p "Enter the end line number: " end

read -p "Enter the file name: " filename

# Call the function to read lines from the file

read\_lines $start $end "$filename"

./5.sh

Enter the start line number: 20

Enter the end line number: 30

Enter the file name: text.txt

20

21

22

23

24

25

26

27

28

29

30

pooya@DESKTOP-741TI2M:/mnt/c/Users/shadi/Desktop/3$ ./5.sh

Enter the start line number: 60

Enter the end line number: 65

Enter the file name: text.txt

60

61

62

63

64

65

بخش آخر:

# !/bin/bash

function shape\_1(){

    for i in {1..5}

    do

        for j in `seq $i`

        do

            echo -n $i

        done

        echo

    done

}

function shape\_2(){

    for i in {1..6}

    do

        ((num\_of\_space = 6-$i))

        for sapce in `seq $num\_of\_space`

        do

            echo -n " "

        done

        for j in `seq $i`

        do

            echo -n ". "

        done

        echo

    done

    for i in {6..1..-1}

    do

        ((num\_of\_space = 6-$i))

        for sapce in `seq $num\_of\_space`

        do

            echo -n " "

        done

        for j in `seq $i`

        do

            echo -n ". "

        done

        echo

    done

}

function shape\_3(){

    for i in {1..5}

    do

        ((num\_of\_line = $i-1))

        for j in `seq $num\_of\_line`

        do

            echo -n "| "

        done

        echo "|\_"

    done

}

echo "Enter number (1, 2, 3): "

read n

case $n in

1)shape\_1

;;

2)shape\_2

;;

3)shape\_3

;;

\*)echo "Error: enter a valid input!"

;;

esac

./6.sh

Enter number (1, 2, 3):

2

.

. .

. . .

. . . .

. . . . .

. . . . . .

. . . . . .

. . . . .

. . . .

. . .

. .

.

pooya@DESKTOP-741TI2M:/mnt/c/Users/shadi/Desktop/3$ ./6.sh

Enter number (1, 2, 3):

3

|\_

| |\_

| | |\_

| | | |\_

| | | | |\_

pooya@DESKTOP-741TI2M:/mnt/c/Users/shadi/Desktop/3$