test

????????

```
Str1 = str2 | 當str1與str2相同時, 傳回True
Str1 != str2| 當str1與str2不同時, 傳回True
Str1 < Str2
Str1 <= Str2
Str1 > Str2
Str1 >= Str2
Str     | 當str不是空字符時, 傳回True
-n str  | 當str的長度大於0時, 傳回True
-z str  | 當str的長度是0時, 傳回True
```

??(??) ??

```
Int1 -eq int2 |當int1等於int2時, 傳回True
Int1 -ge int2 |當int1大於/等於int2時, 傳回True
Int1 -le int2 |當int1小於/等於int2時, 傳回True
Int1 -gt int2 |當int1大於int2時, 傳回True
Int1 -ne int2 |當int1不等於int2時, 傳回True
Int1 -lt int2 |當int1小於 int2時, 傳回True
```

?????

```
-e file   | 檔案是否存在
-d file   | 當file是一個目錄時, 傳回 True
-f file   | 當file是一個普通檔案時, 傳回 True
-r file   | 當file是一個可讀檔案時, 傳回 True
-s file   | 當file檔案內容長度大於0時, 傳回 True; 空白內容傳回 False
-w file   | 當file是一個可寫檔案時, 傳回 True
-x file   | 當file是一個可執行檔案時, 傳回 True
```

File operators list

Operator	Returns

`-a FILE`	True if file exists.
`-b FILE`	True if file is block special.
`-c FILE`	True if file is character special.
`-d FILE`	True if file is a directory.
`-e FILE`	True if file exists.
`-f FILE`	True if file exists and is a regular file.
`-g FILE`	True if file is set-group-id.
`-h FILE`	True if file is a symbolic link.
`-L FILE`	True if file is a symbolic link.
`-k FILE`	True if file has its `sticky' bit set.
`-p FILE`	True if file is a named pipe.
`-r FILE`	True if file is readable by you.
`-s FILE`	True if file exists and is not empty.
`-S FILE`	True if file is a socket.
`-t FD`	True if FD is opened on a terminal.
`-u FILE`	True if the file is set-user-id.
`-w FILE`	True if the file is writable by you.
`-x FILE`	True if the file is executable by you.
`-O FILE`	True if the file is effectively owned by you.
`-G FILE`	True if the file is effectively owned by your group.
`-N FILE`	True if the file has been modified since it was last read.
`! EXPR`	Logical not.
`EXPR1 && EXPR2`	Perform the `and` operation.
`EXPR1 || EXPR2`	Perform the `or` operation.

??????

```
# Let us Declare Two Boolean Variables
# Set this one to true
jobstatus=true
# Check it
if [ "$jobstatus" = true ] ; then
echo 'Okay :)'
```

```
else
    echo 'Noop :('
fi
# Double bracket format syntax to test Boolean variables in bash
bool=false
if [[ "$bool" = true ]] ; then
    echo 'Done.'
else
    echo 'Failed.'
fi
```

????

```
# 如果前個指令執行有錯誤時，會執行兩個程序
[ ! $? -eq 0 ] && echo "Abort the process!! you can try it agian after you made change." && exit 1

# 或者
[ ! $? -eq 0 ] && {
 echo "Abort the process!! you can try it agian after you made change."
 exit 1
}

#
if [[ $RETURN_CODE != 0 ]]; then
        echo "Zulip first start database initi failed in \"initialize-database\" exit code $RETURN_CODE. Exiting."
        exit $RETURN_CODE
fi

#
if [[ $TIMEOUT -eq 0 ]]; then
            echo "Could not connect to database server. Exiting."
            unset PGPASSWORD
            exit 1
fi
#
if (($? > 0)); then
            echo "$SECRET_KEY = $SECRET_VAR" >> "$DATA_DIR/zulip-secrets.conf"
            echo "Secret added for \"$SECRET_KEY\"."
fi
#
if test "x$newbranch" = x; then
```

```
    newbranch=`git branch -a | grep "*" | cut -d ' ' -f2`
fi

# Check router home directory.
[ -d "$PROD_HOME" ] || {
echo "Router home directory ($PROD_HOME) not found"
exit 1
}

# AND
autoBackupConfiguration() {
    if ([ "$AUTO_BACKUP_ENABLED" != "True" ] && [ "$AUTO_BACKUP_ENABLED" != "true" ]); then
        rm -f /etc/cron.d/autobackup
        echo "Auto backup is disabled. Continuing."
        return 0
    fi
}

# OR
if [ "$MANUAL_CONFIGURATION" = "False" ] || [ "$MANUAL_CONFIGURATION" = "false" ]; then
        databaseConfiguration
        secretsConfiguration
        authenticationBackends
        zulipConfiguration
fi

# Die if $f1 or $f2 is missing
if [ ! -f "$f1"  ] ||  [ ! -f "$f2" ]
then
echo "Required files are missing."
else
echo "Let us build SFTP jail."
fi
# And
if [[ $age -ge 18 ]] && [[ $nat -eq "Indian" ]];then
    echo "You can vote!!!"
else
    echo "You can not vote"

fi
```

```
# multiple AND
if [ -e "$DATA_DIR/.initiated" ] && ([ "$FORCE_FIRST_START_INIT" != "True" ] && [ "$FORCE_FIRST_START_INIT"
!= "true" ]); then
    echo "First Start Init not needed. Continuing."
    return 0
fi



# one-liner
 [[ -z "$var" ]] && echo "NULL" || echo "NOT NULL"
# if age is greater than or equal to 18 then the program will print "Adult" or it will print "Minor".
[[ $age -ge 18 ]] && echo "Adult" || echo "Minor"

# Contain a substring
if [[ $var = *pattern1* ]]; then
    echo "Do something"
fi

[[ $1 != *cyberciti.biz/faq/* ]] && { printf "Error: Specify faq url (e.g., http://www.cyberciti.biz/faq/url-1-2-3/)\n";
exit 2; }

if [[ $fullstring == *"$substr"* ]];

# Regex
if [[ $ip =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]]; then

head="win00000:hascom_command.ksh : hostname:wintstc, monitor port#:17911"
if [[ $head =~ ^win00000.*$ ]]; then

# Check the the number of the version
[ "$(echo "$TMUX_VERSION >= 2.4" | bc)" = 1 ] || echo "The version $TMUX_VERSION is outdated"
```

Revision #14
Created 1 June 2020 06:26:02 by Admin
Updated 22 October 2024 16:22:19 by Admin