#!/usr/bin/env bash

# Use this script to test if a given TCP host/port are available

cmdname=$(basename $0)

#shell中0表示标准输入，1表示标准输出，2表示标准错误输出，>默认为标准输出重定向

#2>$1 把标准错误输出重定向到标准输出 &>file 把标准输出和标准错误输出都重定向到文件file中

echoerr() { if [[ $QUIET -ne 1 ]]; then echo "$@" 1>&2; fi }

usage()

{

cat << USAGE >&2

Usage:

$cmdname host:port [-s] [-t timeout] [-- command args]

-h HOST | --host=HOST Host or IP under test

-p PORT | --port=PORT TCP port under test

Alternatively, you specify the host and port as host:port

-s | --strict Only execute subcommand if the test succeeds

-q | --quiet Don't output any status messages

-t TIMEOUT | --timeout=TIMEOUT

Timeout in seconds, zero for no timeout

-- COMMAND ARGS Execute command with args after the test finishes

USAGE

exit 1

}

wait\_for()

{

if [[ $TIMEOUT -gt 0 ]]; then

echoerr "$cmdname: waiting $TIMEOUT seconds for $HOST:$PORT"

else

echoerr "$cmdname: waiting for $HOST:$PORT without a timeout"

fi

start\_ts=$(date +%s)

while :

do

(echo > /dev/tcp/$HOST/$PORT) >/dev/null 2>&1

#尝试连接:host 这台机器，对应port端口

#在/dev/tcp伪设备上执行echo命令，/dev/tcp会尝试进行tcp连接，然后把输出结果重定向到/dev/null上，即不保存输出结果。2>&1表示将错误输出重定向到标准输出上。

result=$?

#上一个指令的返回值，0表示连接成功，其他值表示有错误

if [[ $result -eq 0 ]]; then

end\_ts=$(date +%s) #将当前时间转化为秒

echoerr "$cmdname: $HOST:$PORT is available after $((end\_ts - start\_ts)) seconds"

break

fi

sleep 1

done

return $result

}

wait\_for\_wrapper()

{

# In order to support SIGINT during timeout: http://unix.stackexchange.com/a/57692

if [[ $QUIET -eq 1 ]]; then

timeout $TIMEOUT $0 --quiet --child --host=$HOST --port=$PORT --timeout=$TIMEOUT &

#命令后面加&表示设置此进程为后台进程

else

timeout $TIMEOUT $0 --child --host=$HOST --port=$PORT --timeout=$TIMEOUT & #起一个子进程 timeout $TIMEOUT 表示设定时间

fi

PID=$!

# $!表示Shell最后运行的后台Process的PID

trap "kill -INT -$PID" INT

wait $PID

RESULT=$?

# $?表示最后运行的命令的结束代码（返回值）

if [[ $RESULT -ne 0 ]]; then

echoerr "$cmdname: timeout occurred after waiting $TIMEOUT seconds for $HOST:$PORT"

fi

return $RESULT

}

# process arguments

while [[ $# -gt 0 ]] #接收用户传递的参数 $#表示添加到Shell的参数个数

do

case "$1" in

\*:\* )

hostport=(${1//:/ })

HOST=${hostport[0]}

PORT=${hostport[1]}

shift 1

;;

--child)

CHILD=1

shift 1

;;

-q | --quiet)

QUIET=1

shift 1

;;

-s | --strict)

STRICT=1

shift 1

;;

-h)

HOST="$2"

if [[ $HOST == "" ]]; then break; fi

shift 2

;;

--host=\*)

HOST="${1#\*=}"

shift 1

;;

-p)

PORT="$2"

if [[ $PORT == "" ]]; then break; fi

shift 2

;;

--port=\*)

PORT="${1#\*=}"

shift 1

;;

-t)

TIMEOUT="$2"

if [[ $TIMEOUT == "" ]]; then break; fi

shift 2

;;

--timeout=\*)

TIMEOUT="${1#\*=}"

shift 1

;;

--)

shift

CLI="$@" # $@表示所有参数列表

break

;;

--help)

usage

;;

\*)

echoerr "Unknown argument: $1"

usage

;;

esac

done

if [[ "$HOST" == "" || "$PORT" == "" ]]; then

echoerr "Error: you need to provide a host and port to test."

usage

fi

TIMEOUT=${TIMEOUT:-15}

STRICT=${STRICT:-0}

CHILD=${CHILD:-0}

QUIET=${QUIET:-0}

if [[ $CHILD -gt 0 ]]; then

wait\_for

RESULT=$?

exit $RESULT

else

if [[ $TIMEOUT -gt 0 ]]; then

wait\_for\_wrapper

RESULT=$?

else

wait\_for

RESULT=$?

fi

fi

if [[ $CLI != "" ]]; then

if [[ $RESULT -ne 0 && $STRICT -eq 1 ]]; then

echoerr "$cmdname: strict mode, refusing to execute subprocess"

exit $RESULT

fi

exec $CLI

else

exit $RESULT

fi

**Shell笔记：**

1. Shell重定向＆>file、2>&1、1>&2的区别

0表示标准输入；1表示标准输出；2表示标准错误输出；>默认为标准输出重定向，与1>相同；

2>&1 意思是把 标准错误输出 重定向到 标准输出.&>file 意思是把 标准输出 和 标准错误输出 都重定向到文件file中