#!/bin/sh

#

##############################################################################

##############################################################################

### This script was written to kill autosys jobs on remote machines, using

### parameters passed in from the command line. There are 7 parameters that

### can be passed, and they are:

###

### -a Kill all processes found if more than 1 is found.

### The default is false.

### Acceptible values are: "t", "true"

###

### -d The directory in which to put the log files.

### The default is "/tmp", unless you include this parm

### on the command line, then it is "/usr/local/ccms/exe".

###

### -f Success/failure if the job cannot be found.

### The default is failure.

### Acceptible values are: "s","success"

###

### -g The list of strings to grep for to find a particular

### job(s) to be killed. There is no default value and the job

### will fail without this parameter. This field is space

### delimited, and all of the strings found here will be

### greped for with the "and" condition;

### i.e. grep strnga | grep strngb ...

###

### -k Success/Failure if the job(s) cannot be killed.

### The default is failure.

### Acceptible values are: "s","success"

###

### -p PID. Numeric process ID of the job to be killed. Either the -p

### or the -g parameter (but not both) must be specified.

###

### -s List of signals to be sent to kill a job. There is

### no default, and numbers or characters will be accepted.

### There MUST be at least one value for this parameter,

### the job will fail without at least one.

###

### -t Timeout. The length of time, in minutes, between

### sending subsequent signals from the list of signals.

### The default value is 5.

###

### The syntax for using this command is as follows, the "[ ]" imply

### that these parameters are optional, the "..." imply as many more

### values as you want to enter:

###

### nak -g root sybase ... -s 1 1 1 2 3 15 9 ...

### [-a t] [-t 25] [-k s] [-f s] [-d]

###

### The following example will look for the single string

### "root sybase", the above example will look for two strings,

### "root" and "sybase".

###

### nak -g "root sybase" ... -s 1 1 1 2 3 15 9 ...

### [-a t] [-t 25] [-k s] [-f s] [-d]

###

### The following example will look for the process with

### Process ID equal to 12345.

###

### nak -p "12345" -s 1 1 1 2 3 15 9 ...

### [-a t] [-t 25] [-k s] [-f s] [-d]

###

### The following is a list of the return codes and their meaning:

###

### 0 -- Denotes success in all cases.

###

### 99 -- Denotes a syntax error.

###

### 98 -- Failure when multiple jobs are found, and "killall" = false.

###

### 97 -- Failure if the job(s) is not found.

###

### 96 -- Failure to kill all jobs, whether it be one or multiple jobs.

###

### 95 -- Failure of the kill command, probably a bad signal.

###

### \*\*Note: Processes that contain the words "grep" and "nak" are excluded

### from the list of processes that can be killed. This is done to to keep

### the script "nak" from killing its self, and from trying to kill

### all of the grep processes that it spawns.

###

###

### Author: Michael Ryan Created: 2/03/97

### James Perchik

###

### Revised: Date Comments

###

###

##############################################################################

##############################################################################

gstrng="" #List of strings to grep/search for.

list="" #List of successive signals to be sent.

timelmt=5 #Amount of time between sending signals.

killall="false" #Kill all processes if multiple are found.

notfound="failure" #What to return if job is not found.

notkilled="failure" #What to return if job could not be killed.

first=""

rest=""

alldone="false"

tlist="/tmp/autosys.list."

tpids="/tmp/autosys.pids."

logdir="/tmp"

logfile="autosys.kill.log."$$

log="$logdir""/""$logfile"

rmlog="autosys.kill.log.\*"

sintax1="You must include \"-g\" and \"-s\" parms w/ 1 or more values each."

sintax2="Syntax: nak { -g string ... | -p pid } -s sig ... [-k s] [-t 5] [-a t] [-f s] [-d]"

exec 2>&1

###############################################################

#### Process all of the input parameters. ####

###############################################################

while [ x`echo $1 | cut -b 1` = "x-" ]

do

case $1 in

-a) #Kill all if multiple jobs are found? (fail = 98)

shift

if [ $1 = "true" ] || [ $1 = "t" ]

then killall="true"

else

echo "bad value for the killall option, use true or t"

echo " RC = 99"

echo $sintax2

echo "bad value for the killall option, use true or t" >> $log

echo " RC = 99" >> $log

exit 99

fi

shift ;;

-d) #Use this as a flag to set the dir to /usr/local/ccms/exe

logdir="/usr/local/ccms/exe"

log="$logdir""/""$logfile"

shift ;;

-f) #Return Success or Failure if job not found. (fail=97)

shift

if [ $1 = "success" ] || [ $1 = "s" ]

then notfound="success"

else

echo "bad value for the found option, use success or s"

echo "RC = 99"

echo $sintax2

echo "bad value for the found option, use success or s" >> $log

echo "RC = 99" >> $log

exit 99

fi

shift ;;

-g) #Build grep string.

shift

case $1 in

-\* ) ;; #Jump out if next parm

\* )

while [ x"$1" != x ]

do

gstrng="$gstrng""grep "\"$1\"

shift

case $1 in

-\* | "" ) break ;; #Jump out of while @ next parm

esac

gstrng="$gstrng"" | "

done;;

esac

gstrng="$gstrng"" | grep -v \" grep \" | grep -v \"[/| ]nak \""

;;

-p) #Search for numeric PID

shift

case $1 in

-\* | "" )

echo "no pid value specified for -p option"

echo "RC = 99"

echo $sintax2

echo "no pid value specified for -p option" >> $log

echo "RC = 99" >> $log

exit 99

;;

esac

gstrng="grep \"^ \*[^ ][^ ]\*[ ][ ]\*$1\""

shift ;;

-k) #Return Success or Failure if job can't be killed. (fail=96)

shift

if [ $1 = "success" ] || [ $1 = "s" ]

then notkilled="success"

else

echo "bad value for the notkilled option, use success or s"

echo "RC = 99"

echo $sintax2

echo "bad value for the notkilled option, use success or s" >> $log

echo "RC = 99" >> $log

exit 99

fi

shift ;;

-s) #List of signals to send.

shift

case $1 in

-\* ) ;;

\* )

while [ x"$1" != x ]

do

list="$list"" $1"

shift

case $1 in

-\* ) break ;;

esac

done;;

esac;;

-t) #Time between sending signals.

shift

timelmt=$1

shift ;;

-\*) #Reject anything that is not one of the above parms.

echo $sintax1

echo $sintax2

echo "RC = 99"

echo "Invalid usage, check the syntax and try again." >> $log

echo "RC = 99" >> $log

exit 99

esac

done

if [ x"$gstrng" = x ] || [ x"$list" = x ]

then

echo $sintax1

echo $sintax2

echo "RC = 99"

echo $sintax1 >> $log

echo $sintax2 >> $log

echo "RC = 99" >> $log

exit 99

fi

echo "The grep string is:"

echo $gstrng

echo "The string of signals is:"

echo $list

echo "The grep string is:" >> $log

echo $gstrng >> $log

echo "The string of signals is:" >> $log

echo $list >> $log

find /tmp -name "$rmlog" -ctime +7 -exec rm -f {} \; >> $log 2>&1

#Remove any log files over 7 days old, user created.

find /usr/local/ccms/exe -name "$rmlog" -ctime +7 -exec rm -f {} \; >> $log 2>&1

#Remove any log files over 7 days old, autosys created.

###############################################################

#### Determine machine type and get the process ids ####

###############################################################

machtype=`uname -a | cut -f 1,3 -d " "`

case $machtype in

"SunOS 4"\* | "ULTRIX"\* ) #Accept only Ultrix and <SunOS 5

pids=`ps -aux |eval $gstrng| awk 'BEGIN {FS=" "} {print $2}'`

echo "These are the process ids found." >> $log

ps -aux |eval $gstrng >> $log

echo "$pids" >> $log

;;

\* ) #Accept all other systems

pids=`ps -fae |eval $gstrng| awk 'BEGIN {FS=" "} {print $2}'`

echo "These are the process ids found." >> $log

ps -fae |eval $gstrng >> $log

echo "$pids" >> $log

;;

esac

###############################################################

#### Determine whether or not to start killing the ####

#### processes found. If we exit here, return success ####

#### or failure, which is pre determined. ####

###############################################################

if [ x`echo $pids | awk 'BEGIN {FS=" "} {print $1}'` = x ]

then

if [ $notfound = "success" ] #No pids found

then

echo "No processes were found."

echo "RC = 0"

echo "No processes were found." >> $log

echo "RC = 0" >> $log

exit 0 #Return success

else

echo "No processes were found."

echo "RC = 97"

echo "No processes were found." >> $log

echo "RC = 97" >> $log

exit 97 #Return failure

fi

elif [ x`echo $pids | awk 'BEGIN {FS=" "} {print $2}'` != x ]

then

if [ $killall = "false" ]

then #Multiple pids found

echo "Multiple processes found, and cannot kill all."

echo "RC = 98"

echo "Multiple processes found, and cannot kill all." >> $log

echo "RC = 98" >> $log

exit 98

fi

fi

###############################################################

#### Start killing processes, using the first signal ####

#### in the list provided. Check to see that all pids ####

#### are dead or the list of signals is used up before ####

#### we leave this loop. ####

###############################################################

slptime=`expr $timelmt \\* 60`

src=$?

if [ $src -gt 1 ]

then

echo $sintax1

echo $sintax2

echo "RC = 99"

echo "Invalid usage, check the syntax and try again." >> $log

echo "RC = 99" >> $log

exit 99

elif [ $src -eq 1 ]

then

slptime=0

fi

templist=$tlist$$

temppids=$tpids$$

while [ $alldone = "false" ]

do

newpids=""

echo $list >$templist

echo $pids >$temppids

read first rest <$templist

echo "trying signal "$first

echo "trying signal "$first >> $log

kill -"$first" $pids

krc=$?

if [ $krc != 0 ]

then

echo "Bad return code from the kill command: $krc" >> $log

echo "Please check the command and try again." >> $log

echo "RC = 95" >> $log

echo "Bad return code from the kill command: $krc"

echo "Please check the command and try again."

echo "RC = 95"

exit 95

fi

polltime=1

sleeptime=$slptime

while [ $sleeptime -gt 0 ]

do

chk=""

case $machtype in

"SunOS 4"\* | "ULTRIX"\* ) #Ultrix and <SunOS 5

chk=`ps -aux | eval $gstrng`

;;

\* ) #All other systems

chk=`ps -fae | eval $gstrng`

;;

esac

if [ x"$chk" != x ]

then

sleep $polltime

case $polltime in

"1" ) polltime=2 ;;

"3" ) polltime=4 ;;

"4" ) polltime=8 ;;

"8" ) polltime=15 ;;

esac

sleeptime=`expr $sleeptime - $polltime`

else

sleeptime=0

fi

done

if [ x"$rest" = x ] #Process list of signals 1 @ a time

then

alldone="true"

else

list="$rest"

fi

read pid1 pid2 <$temppids

while [ x"$pid1" != x ] #process list of pids, drop

do #killed pids from list

chk=""

case $machtype in

"SunOS 4"\* | "ULTRIX"\* ) #Ultrix and <SunOS 5

chk=`ps -aux | grep $pid1 | eval $gstrng`

;;

\* ) #All other systems

chk=`ps -fae | grep $pid1 | eval $gstrng`

;;

esac

if [ x"$chk" != x ]

then

newpids="$newpids ""$pid1"

fi

echo $pid2 >$temppids

read pid1 pid2 <$temppids

done

if [ x"$newpids" = x ]

then

alldone="true"

fi

pids="$newpids"

done

rm $tlist\*

rm $tpids\*

if [ $alldone = "true" ] && [ x"$pids" != x ]

then

echo "Not all the processes were killed."

echo "RC = 96"

echo "Not all the processes were killed." >> $log

echo "RC = 96" >> $log

exit 96

fi

if [ $killall = "true" ]

then

echo "All the processes were killed."

echo "All the processes were killed." >> $log

else

echo "One process was killed."

echo "One process was killed." >> $log

fi

echo "RC = 0"

echo "RC = 0" >> $log

exit 0