#!/bin/sh

#remote shell ecall 远程运行脚本

#Editor: 王习平

#date : 2006-08-26

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

#define area 定义区域

#remote copy command

#CP=scp

CP=`getcfg.sh CP`

#remote shell

SH=`getcfg.sh SH`

#shell argument

SH\_ARG=`getcfg.sh SH\_ARG`

#PATH setup 路径设置

LocalShellPath=`getcfg.sh LocalShellPath`

LocalBasePath=`getcfg.sh LocalBasePath`

LocalObjectPath=`getcfg.sh LocalObjectPath`

LocalConfigPath=`getcfg.sh LocalConfigPath`

LocalTempPath=`getcfg.sh LocalTempPath`

ServiceListFile=`getcfg.sh ServiceListFile`

SanPath=`getcfg.sh SanPath`

BasePath=`getcfg.sh BasePath`

ObjInSanPath=`getcfg.sh ObjInSanPath`

CurrentVersion=`getcfg.sh CurrentVersion`

CheckVersion=`getcfg.sh CheckVersion`

LD\_LIBRARY\_PATH=`getcfg.sh LD\_LIBRARY\_PATH`;export LD\_LIBRARY\_PATH

checkenv()

{

envlist="CP SH LocalShellPath LocalBasePath LocalObjectPath LocalConfigPath LocalTempPath ServiceListFile BasePath CurrentVersion ORACLE\_BASE ORACLE\_HOME NLS\_LANG LD\_LIBRARY\_PATH SHLIB\_PATH OCI\_PATH"

CheckPass="true"

for i in $envlist

do

value=`getcfg.sh $i`

if [ "$value" = "" ]

then

echo Error: $i not set value!

CheckPass="false"

fi

done

if [ "$CheckPass" = "false" ]

then

echo Please set value in sh.cfg or use setcfg set value

return 1

fi

return 0

}

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

#impl area 功能函数区

#用户可以根据实际需要增加 可以通过ecall function来调用function中实现的功能

#ecall会为功能函数传入两个参数 $1:远程机器名 $2:远程服务名 远程机器名=远程服务名+编号 远程机器名定义在/etc/hosts中

#构建服务运行依赖的基本目录

makebasepath()

{

#先删除用户目录下指定的文件夹，再创建

$SH $1 $SH\_ARG rm -rf ${BasePath}$1 > /dev/null

$SH $1 $SH\_ARG mkdir ${BasePath}$1 > /dev/null

if [ "${SanPath}" = "" ]

then

#若无san，则在本地硬盘BasePath创建工作目录

$SH $1 $SH\_ARG mkdir ${BasePath}$1/flow > /dev/null 2> /dev/null

$SH $1 $SH\_ARG mkdir ${BasePath}$1/dump > /dev/null 2> /dev/null

$SH $1 $SH\_ARG mkdir ${BasePath}$1/log > /dev/null 2> /dev/null

else

#如果有san，先删除san目录中的指定文件夹，再创建

$SH $1 $SH\_ARG rm -rf ${SanPath}$1 > /dev/null 2> /dev/null

$SH $1 $SH\_ARG mkdir ${SanPath}$1 > /dev/null 2> /dev/null

#在SanPath中创建子文件夹，用于保存常变文件

$SH $1 $SH\_ARG mkdir ${SanPath}$1/flow > /dev/null 2> /dev/null

$SH $1 $SH\_ARG mkdir ${SanPath}$1/dump > /dev/null 2> /dev/null

$SH $1 $SH\_ARG mkdir ${SanPath}$1/log > /dev/null 2> /dev/null

#建立链接关系

$SH $1 $SH\_ARG ln -s ${SanPath}$1/flow $1/flow 2>/dev/null >/dev/null

$SH $1 $SH\_ARG ln -s ${SanPath}$1/dump $1/dump 2>/dev/null >/dev/null

$SH $1 $SH\_ARG ln -s ${SanPath}$1/log $1/log 2>/dev/null >/dev/null

fi

#创建其它子文件夹，用于保存不常改变的文件

if [ "${ObjInSanPath}" = "true" -a "${SanPath}" != "" ]

then

#若ObjInSanPath值为true且有san，则在SanPath中创建以下文件夹,然后链接到用户目录

$SH $1 $SH\_ARG mkdir ${SanPath}$1/bin > /dev/null 2> /dev/null

$SH $1 $SH\_ARG mkdir ${SanPath}$1/config > /dev/null 2> /dev/null

$SH $1 $SH\_ARG ln -s ${SanPath}$1/bin $1/bin 2>/dev/null >/dev/null

$SH $1 $SH\_ARG ln -s ${SanPath}$1/config $1/config 2>/dev/null >/dev/null

else

#缺省情况下直接在用户目录创建以下文件夹

$SH $1 $SH\_ARG mkdir ${BasePath}$1/config > /dev/null 2> /dev/null

$SH $1 $SH\_ARG mkdir ${BasePath}$1/bin > /dev/null 2> /dev/null

fi

#创建shell文件夹

$SH $1 $SH\_ARG mkdir ${BasePath}shell > /dev/null 2> /dev/null

}

#删除服务器运行依赖的基本文件

delbasepath()

{

if [ "${SanPath}" != "" ]

then

#删除san中的文件夹

$SH $1 $SH\_ARG rm -rf ${SanPath}$1 > /dev/null 2> /dev/null

fi

#只需删除自身文件夹

$SH $1 $SH\_ARG rm -rf ${BasePath}$1 > /dev/null

#删除shell文件夹

rm -f /tmp/sh.rm >/dev/null 2>/dev/null

>/tmp/sh.rm

$SH $1 $SH\_ARG [ -r /tmp/sh.rm ]

if [ $? -ne 0 ]

then

$SH $1 $SH\_ARG rm -rf ${BasePath}shell > /dev/null

fi

}

#start sloglog simplelog

startslog()

{

if [ $1 = "monitor" ]

then

$SH $1 $SH\_ARG ${BasePath}shell/startslog

fi

}

#从工作服务器获得流水文件

getflow()

{

cd $LocalTempPath

mkdir ${BasePath}$1 2>/dev/null

mkdir ${BasePath}$1/flow 2>/dev/null

$CP $1:${BasePath}$1/flow/\*.con $1/flow 2>/dev/null

$CP $1:${BasePath}$1/flow/\*.id $1/flow 2>/dev/null

}

getslog()

{

cd $LocalTempPath

mkdir $1 2>/dev/null

mkdir $1/flow 2>/dev/null

$CP $1:${BasePath}$1/flow/\*.slog $1/flow 2>/dev/null

}

#获取工作服务器dump文件

getdump()

{

cd $LocalTempPath

mkdir $1 2>/dev/null

mkdir $1/dump 2>/dev/null

$CP $1:${BasePath}$1/dump/\* $1/dump 2>/dev/null

}

#获取工作服务器日志文件

getlog()

{

cd $LocalTempPath

mkdir $1 2>/dev/null

mkdir $1/log 2>/dev/null

$CP $1:${BasePath}$1/log/\* $1/log 2>/dev/null

$CP $1:${BasePath}$1/dump/Syslog.log $1/log 2>/dev/null

}

#获取工作服务器core文件

getcore()

{

cd $LocalTempPath

mkdir $1 2>/dev/null

mkdir $1/bin 2>/dev/null

$CP $1:${BasePath}$1/bin/core $1/bin 2>/dev/null

}

#清理工作服务器日志、core、输出文件

clean()

{

$SH $1 $SH\_ARG echo "" ">" ${BasePath}$1/dump/Syslog.log

$SH $1 $SH\_ARG rm ${BasePath}$1/bin/core.\*

$SH $1 $SH\_ARG rm ${BasePath}$1/log/out

$SH $1 $SH\_ARG rm ${BasePath}$1/flow/\*

$SH $1 $SH\_ARG rm ${BasePath}$1/dump/\*

}

#备份——备份排队机流水与交易引擎的日志和dump文件

backup()

{

BackupListFile=`getcfg.sh BackupListFile`

if [ ! -r "$BackupListFile" ]

then

return

fi

export COMPUTER=$1

BackupLists=`cat $BackupListFile|awk '{

computer=ENVIRON["COMPUTER"]

if ( $1 == substr(computer,1,length($1)) || $1 == "all" )

{

for (i=2;i<=NF;i++)

printf "%s ",$i

}

}'|xargs echo`

if [ -n "$BackupLists" ]

then

mkdir $1 2>/dev/null

for f in $BackupLists

do

dir=`dirname $f`

if [ ! -d $1/$dir ]

then

mkdir $1/$dir 2>/dev/null

fi

$CP $1:${BasePath}$1/$f $1/${dir} 2>/dev/null

done

fi

}

setflow()

{

cd $LocalTempPath

$CP $1/flow/\*.con $1:${BasePath}$1/flow 2>/dev/null

$CP $1/flow/\*.id $1:${BasePath}$1/flow 2>/dev/null

}

#发布服务器运行依赖的基本文件(cpall.sh)

cpbase()

{

$CP ${LocalConfigPath}DeployConfig.xml $1:${BasePath}$1/config/

$CP ${LocalConfigPath}SystemConfig.xml $1:${BasePath}$1/config/

$SH $1 $SH\_ARG ">" /tmp/sh.cp

rm /tmp/sh.cp 2>/dev/null

$SH $1 $SH\_ARG [ -r /tmp/sh.cp ]

if [ $? -ne 1 ]

then

$CP ${LocalShellPath}\* $1:${BasePath}shell/

$SH $1 $SH\_ARG rm -f /tmp/sh.cp >/dev/null

fi

No=`getno $1 $2`

if [ $2 = "tkernel" -a -r ${LocalConfigPath}DeployConfig$No.xml ]

then

$CP ${LocalConfigPath}DeployConfig$No.xml $1:${BasePath}$1/config/DeployConfig.xml

fi

if [ -r ${LocalConfigPath}DeployConfig.$2.xml ]

then

$CP ${LocalConfigPath}DeployConfig.$2.xml $1:${BasePath}$1/config/DeployConfig.xml

fi

if [ -r ${LocalConfigPath}DeployConfig.$1.xml ]

then

$CP ${LocalConfigPath}DeployConfig.$1.xml $1:${BasePath}$1/config/DeployConfig.xml

fi

if [ -r ${LocalConfigPath}SystemConfig.$2.xml ]

then

$CP ${LocalConfigPath}SystemConfig.$2.xml $1:${BasePath}$1/config/SystemConfig.xml

fi

if [ -r ${LocalConfigPath}SystemConfig.$1.xml ]

then

$CP ${LocalConfigPath}SystemConfig.$1.xml $1:${BasePath}$1/config/SystemConfig.xml

fi

}

#基本函数区

#base function

#get base name

getbasename()

{

echo $1 |awk '{

tmp\_str=$1

pos=0

for(ret=-1;ret!=0;)

{

ret=index(tmp\_str,".")

if (ret !=0 )

{

pos=ret+1

tmp\_str=substr(tmp\_str,pos)

}

}

if (pos!=0)

printf "%s",substr($1,1,length($1)-length(tmp\_str)-1)

else

printf "%s",$1

}'

}

#get expanded name

getexpname()

{

echo $1 |awk '{

tmp\_str=$1

pos=0

for(ret=-1;ret!=0;)

{

ret=index(tmp\_str,".")

if (ret !=0 )

{

pos=ret+1

tmp\_str=substr(tmp\_str,pos)

}

}

if (pos!=0)

printf "%s",tmp\_str

else

printf ""

}'

}

#取子字串

substring()

{

if [ $# -lt 1 ]

then

printf ""

return 0

fi

echo $\* |awk ' {

if ( NF > 1 )

pos=$2

else

pos=1

if ( NF > 2 )

len=$3

else

len=length($1)

if ( len > length($1) )

len=length($1)

if ( pos < 1 )

len=1

printf "%s",substr($1,pos,len)

}'

return 0

}

#从变量list中取出后部分变量

filter()

{

if [ $# -lt 2 ]

then

printf ""

return 0

fi

echo $\* |awk ' {

i=$1

for ( ;i<NF;i++)

printf "%s ",$i

if ( i==NF )

printf "%s",$i

}'

return 0

}

#取出系列变量引用值

reference()

{

for i in $\*

do

first=`substring $i 1 1`

if [ "$first" = "@" ]

then

remain=`substring $i 2`

getcfg.sh $remain

printf " "

else

printf "%s " $i

fi

done

}

getno()

{

echo $1 $2 |awk '{printf "%s",substr("$1",length("$2")+1,1)}'

}

#今天

today()

{

date +%C%y%m%d

}

#明天

tomorrow()

{

year=`date +%C%y`

month=`date +%m`

day=`date +%d`

\_flag="false"

if [ `expr $year % 4` = "0" ]

then

if [ `expr $year % 100` = "0" ]

then

\_flag="false"

else

\_flag="true"

fi

fi

if [ `expr $year % 400` = "0" ]

then

\_flag="true"

fi

day=`expr $day + 1`

if [ $day = "32" ]

then

day="01"

month=`expr $month + 1`

fi

if [ $day = "31" ]

then

if [ $month = "04" -o $month = "06" -o $month = "09" -o $month = "11" ]

then

month=`expr $month + 1`

day="01"

fi

fi

if [ $day = "30" -a $month = "02" ]

then

month="03"

day="01"

fi

if [ $day = "29" -a $month = "02" -a $\_flag = "false" ]

then

month="03"

day="01"

fi

if [ $month = "13" ]

then

month="01"

year=`expr $year + 1`

fi

printf "%04s%02s%02s" $year $month $day

}

#检查远程服务器是否正常

checkalive()

{

rm -f .alive

case `uname -s` in

"HP-UX") ping $1 -n 1 -m 2 >/dev/null || > .alive;;

"Linux") ping $1 -c 1 -w 2 >/dev/null || > .alive;;

"AIX") ping -c 1 -w 2 $1 >/dev/null || > .alive;;

\*) echo OS error!;;

esac

if [ -r .alive ]

then

echo "Warnning:" $1 "not alive"

return 1

fi

return 0

}

#检查配置文件是否被改变

checkchange()

{

rm -f .changed

$CP $1:${BasePath}$1/bin/$2.ini\* /tmp >/dev/null 2>/dev/null

${LocalShellPath}GenMD5.sh -c /tmp/$2.ini >/dev/null || >.changed

if [ -r .changed ]

then

echo Warnning: $1:${BasePath}$2.ini had been changed!

return 1

fi

rm -f /tmp/$2.ini /tmp/$2.ini.md5 >/dev/null 2>/dev/null

return 0

}

#检查是否已经运行

checkrun()

{

rm -f .run

$SH $1 $SH\_ARG ps -ef|grep -wE "$2 \*$3" |grep $LOGNAME|grep -v grep|grep -v "$SH"|grep -v start|grep -v restart|grep -v vi|grep -v "ps -ef" >/dev/null && > .run

if [ -r .run ]

then

echo "Warnning: service already running"

return 1

fi

return 0

}

#检查版本

checkversion()

{

if [ "$CheckVersion" = "true" ]

then

$SH $1 $SH\_ARG "export LD\_LIBRARY\_PATH=${LD\_LIBRARY\_PATH} && cd ${BasePath}$1/bin/ && ./$2 -v" |read tmpvar1 tmpvar2 RemoteVersion

tmpPath=`pwd`

cd ${LocalObjectPath}${CurrentVersion}/$2/

$2 -v |read tmpvar1 tmpvar2 LocalVersion

cd $tmpPath

if [ "$RemoteVersion" != "$LocalVersion" ]

then

echo "Warnning: service version invalid!"

printf "RemoteVersion=[%s] LocalVersion=[%s]\n" $RemoteVersion $LocalVersion

return 1

fi

fi

return 0

}

#启动服务

startservice()

{

checkchange $\*

if [ $? -eq 1 ]

then

return

fi

checkrun $\*

if [ $? -eq 1 ]

then

return

fi

checkversion $\*

if [ $? -eq 1 ]

then

return

fi

#第一个参数是主机名，后面是进程启动参数

svr=$1

#右移一位，滤掉主机名

shift 1

$SH $svr $SH\_ARG ${BasePath}shell/start.sh $\* ">" $svr/log/out &

}

#重新启动服务—保留流文件

restart()

{

checkchange $\*

if [ $? -eq 1 ]

then

return

fi

checkrun $\*

if [ $? -eq 1 ]

then

return

fi

checkversion $\*

if [ $? -eq 1 ]

then

return

fi

#第一个参数是主机名，后面是进程启动参数

svr=$1

#右移一位，滤掉主机名

shift 1

$SH $svr $SH\_ARG ${BasePath}shell/restart.sh $\* ">" $svr/log/out &

}

#停止服务

stopservice()

{

if [ $2 != "monitor" ]

then

$SH $1 $SH\_ARG ${BasePath}shell/stop.sh $2 $3

fi

}

#停止监控探针

stopprobe()

{

$SH $1 $SH\_ARG ${BasePath}shell/stop.sh probe

}

#启动监控探针

startprobe()

{

if [ $2 = "fibproxy" ]

then

return

fi

if [ $2 = "fibgate" ]

then

return

fi

if [ $2 = "monitor" ]

then

return

fi

rm -f .prorun

$SH $1 $SH\_ARG ps -ef|grep probe |grep -v grep|grep -v remsh|grep -v start|grep -v vi >/dev/null && > .prorun

if [ -r .prorun ]

then

echo "Warnning: probe already running"

else

$SH $1 $SH\_ARG ${BasePath}shell/startprobe.sh ">" ./probe/log/out &

fi

}

#空服务，用来测试远程服务配置状况

null()

{

disp="off"

}

#发布运行码

cpobj()

{

objlists=`ls ${LocalObjectPath}${CurrentVersion}/$2/`

for f in $objlists

do

expname=`getexpname $f`

if [ ! "$expname" = "ini" -a ! "$expname" = "md5" ]

then

$CP ${LocalObjectPath}${CurrentVersion}/$2/$f $1:${BasePath}$1/bin

fi

done

}

#发布运行配置文件

cpini()

{

if [ -r ${LocalObjectPath}${CurrentVersion}/$2/$1.ini ]

then

$LocalShellPath/GenMD5.sh -g $LocalObjectPath${CurrentVersion}/$2/$1.ini >/dev/null

$CP ${LocalObjectPath}${CurrentVersion}/$2/$1.ini $1:${BasePath}$1/bin/$2.ini

$CP ${LocalObjectPath}${CurrentVersion}/$2/$1.ini.md5 $1:${BasePath}$1/bin/$2.ini.md5

else

if [ -r ${LocalObjectPath}${CurrentVersion}/$2/$2.ini ]

then

$LocalShellPath/GenMD5.sh -g $LocalObjectPath${CurrentVersion}/$2/$2.ini >/dev/null

$CP ${LocalObjectPath}${CurrentVersion}/$2/$2.ini\* $1:${BasePath}$1/bin/

$CP ${LocalObjectPath}${CurrentVersion}/$2/$2.ini.md5 $1:${BasePath}$1/bin/$2.ini.md5

fi

fi

}

#显示服务进程运行情况

show()

{

$SH $1 $SH\_ARG ps -ef|grep -wE "$2 \*$3"|grep $LOGNAME|grep -v grep|grep -v "$SH"|grep -v "scp "|grep -v "sh " |grep -v start.sh|grep -v vi|grep -v "ps -ef" |grep -v $$|awk -F" " -v Item=$1 'BEGIN { i=0 }

{

if ( i>0 )

printf "\n "

printf " %-16s: %8s %-16s",Item,$5,$8

for ( j=9;j<=NF;j++)

printf "%-8s",$j

i++

}

END {

if ( i <= 0 )

{

printf " %-16s: is \033[7moffline\033[0m",Item

}

}'

printf "\n"

}

#功能函数调用接口

#callImpl i implfunction service no parms

callImpl()

{

if [ $# -eq 3 ]

then

computer=$3

service=$3

else

computer=$3$4

service=$3

fi

if [ $3 = "tinit" ]

then

computer=$3

service=$3

fi

func=$2

no=$1

shift 3

args=`reference $\*`

if [ $func != "show" ]

then

printf "No.%02d %s%-15s:[ %s %s]\n" $no "\_\_\_\_\_\_\_\_\_" $computer $func "$args"

else

printf "%2d " $no

fi

checkalive $computer

if [ $? -eq 1 ]

then

return

fi

$func $computer $service $args

}

#根据远程服务器列表远程调用功能函数

#callall callimplfunction

callall()

{

preName=`date |awk '{print $5}'`

case $# in

1)

grep -v "^#" $ServiceListFile|grep -v "^$" > /tmp/$preName.list;;

2)

if [ $2 = "-n" ]

then

grep -v "^#" $ServiceListFile|grep -v "^$"> /tmp/$preName.list

else

grep -v "^#" $ServiceListFile|grep -v "^$"|grep -w $2 > /tmp/$preName.list

fi;;

3)

if [ $2 = "-n" ]

then

grep -v "^#" $ServiceListFile|grep -v "^$"|grep -v $3 > /tmp/$preName.list

else

grep -v "^#" $ServiceListFile|grep -v "^$"|grep -wE "$2[ | ]\*$3" > /tmp/$preName.list

fi;;

esac

cntexpr=`wc -l /tmp/$preName.list |awk '{print $1}'`

cnt=`expr $cntexpr`

i=0

while [ $i -lt $cnt ]

do

i=`expr $i + 1 `

read srv

callImpl $i $1 $srv

done < /tmp/$preName.list

rm /tmp/$preName.list >/dev/null 2>/dev/null

}

#ecall主调用入口

#eall callimplfunction

ecall()

{

if [ $# -eq 0 ]

then

echo "Usage: $0 cpobj|cpini|show|stopservice|makebasepath|delbasepath|cpbase|backup|startservice|restart|null [filter]"

echo " $0 -c function"

else

if [ $1 = "-c" ]

then

argv=`filter 4 $\*`

$2 $argv

return $?

fi

checkenv

if [ $? -eq 1 ]

then

return

fi

callall $\*

fi

}

ecall $\*