

Solaris FC

Article Number: 727 | Rating: 5/5 from 8 votes | Last Updated: Tue, Jan 11, 2022 8:14 AM

```
#!/bin/bash
# set -e ---> It is very good error detection but sometimes cause of unexpected results for me
# do not REMOVE the any temp files while script is RUNNING !!
# thanks to ( @Perderabo unix.com ) for some perl code ( perlcheck .. )
# /* unix.com @ygemici */

# some needed works for more than call the same function !!! :s
#####

function check_tools() {
id="/usr/bin/id"
if [ ! -x "$id" ] ; then echo "ID check cannot execute !! " ; exit 1 ; fi

nawk="/usr/bin/nawk" #; [ -z "$nawk" ] && nawk=`which nawk >/dev/null 2>&1`
if [ ! -x "$nawk" ] ; then echo "nawk cannot execute !! " ; exit 1 ; fi
#awk=/usr/xpg4/bin/awk

sed="/usr/xpg4/bin/sed"
if [ ! -x "$sed" ] ; then echo "sed cannot be found !! " ; exit 1 ; fi

grep="/usr/xpg4/bin/grep"
if [ ! -x "$grep" ] ; then echo "grep cannot be found !! " ; exit 1 ; fi
#grep=/usr/sfw/bin/gegrep
}

function arraycount() {
a=( $@ )
for((i=0;i<=${#a[@]};i++))
do
sum=$((sum+${#a[i]}))
done
}

function blinked() {
```

```

sleeptest
[ $? -eq 0 ] && sleepx="sleep 0.5" || sleepx="sleep 1"
time=0
#arraycount $@
while ;; do
if [ $time -lt 5 ] ; then
echo -en "\033[5m$@\033[0m"
$sleepx
# clear the last line from term
echo -en "\033[1Kr"

## Clear terminal ##
#echo -en "r"
#printf "\033c"
#printf "ec"
#echo -en "ec"
#echo -e '\033\143'
#printf "%${sum}sr" " "

$sleepx
else
break
fi
((time++))
done
}

function sleeptest() {
sleep 0.2 2>/dev/null
}

function menu_clearer() {
#printf "r"
cnt=$1;x=$2
((sleeptestc++))
if [ $sleeptestc -eq 1 ] ; then
sleeptest
[ $? -eq 0 ] && sleepx="sleep 0.2" || sleepx="sleep 1"

```

```
fi
for((i=0;i<=$cnt;i++))
do
$sleepx
printf "%${x}cr" " "
done
}
```

```
function loopx() {
echo -en "$1"
```

```
if [ -z $2 ] ; then
PID=$!
x='.'
lenl=1
while kill -0 $PID 2>/dev/null ;
#while ps -p $PID >/dev/null ;
do
for((i=0;i<$lenl;i++)); do printf "%s" "$x"
sleep 1
done
done
echo -e "\nJob Completed !!\033[1m [OK]\n"
```

```
else
PID=$2
x=1
while kill -0 $PID 2>/dev/null ;
do
#lenl=12
#printf "Please waitr"
#while [ $x -lt $lenl ]
#do
menu_clearer 1 $x
((x++))
#done
done
```

```
fi
}
```

```
function define_var() {
((define_var++))
if [ $define_var -eq 1 ]; then
f1=${LOCAL_TEMP_DIR}/disksinfos
f2=${LOCAL_TEMP_DIR}/disksinfos_full
f3=${LOCAL_TEMP_DIR}/disksinfos_full_labeln
f5=${LOCAL_TEMP_DIR}/fc_ports
f6=${LOCAL_TEMP_DIR}/fc_controllers
f7=${LOCAL_TEMP_DIR}/diskpaths
f8=${LOCAL_TEMP_DIR}/mpathdisks
f9=${LOCAL_TEMP_DIR}/formatdisks
f10=${LOCAL_TEMP_DIR}/hba_pci_lst
f11=${LOCAL_TEMP_DIR}/diskports_paths
f12=${LOCAL_TEMP_DIR}/diskports_with_cfgadm
f13=${LOCAL_TEMP_DIR}/diskerrors
f14=${LOCAL_TEMP_DIR}/diffs
f15=${LOCAL_TEMP_DIR}/fullformatlabels
f16=${LOCAL_TEMP_DIR}/hba_states
f17=${LOCAL_TEMP_DIR}/all_hba_ports
f18=${LOCAL_TEMP_DIR}/inactive_ports
f19=${LOCAL_TEMP_DIR}/hbagroups_ports
f20=${LOCAL_TEMP_DIR}/diskhbaports
f21=${LOCAL_TEMP_DIR}/fulldiskhbaports
f22=${LOCAL_TEMP_DIR}/fulldiskerrors
f23=${LOCAL_TEMP_DIR}/fulldiskerrors_labeln
f24=${LOCAL_TEMP_DIR}/fulldiskcfgs
f25=${LOCAL_TEMP_DIR}/fulldiskcfgs_labeln
f26=${LOCAL_TEMP_DIR}/fullfull
f27=${LOCAL_TEMP_DIR}/fullfullout
f28=${LOCAL_TEMP_DIR}/inqsymwwn
f29=${LOCAL_TEMP_DIR}/inqsymwwnlabel
f30=${LOCAL_TEMP_DIR}/inqvendor
f31=${LOCAL_TEMP_DIR}/inqvendorlabel
f32=${LOCAL_TEMP_DIR}/inqhba
f33=${LOCAL_TEMP_DIR}/fullmpath
```

```

f35=${LOCAL_TEMP_DIR}/iostats_ssd
f36=${LOCAL_TEMP_DIR}/hba_cfg
f37=${LOCAL_TEMP_DIR}/cfg_pci
f38=${LOCAL_TEMP_DIR}/cfg_pci_labels
f39=${LOCAL_TEMP_DIR}/targetwwns
f41=${LOCAL_TEMP_DIR}/hba_pci
f42=${LOCAL_TEMP_DIR}/list_luns
f43=${LOCAL_TEMP_DIR}/disk_mpaths
fi
}

```

```

function find_pci_addr() {
ls -l "$1"|$sed 's/.*devices(.*)/fp.*1/'
}

```

```

function detect_pci_inf() {
f4=${LOCAL_TEMP_DIR}/pci_sparc_tx_x
f4_clone=${f4}_clone
if [ -x /usr/sbin/prtdiag ] ; then
prtdiagnawk '/IO Devices/{f=1;print $0 ;next}/^$/{f=0;}f;' > $f4 &
loopx "Detecting system platform PCI devices "
fi
nawk '$1~/PCI/{if(p==$1||!p){x=$1;getline;sub("/.*$ ", "", $1);a[x]=a[x]FS$1;}else p=$1}
END{for(i in a){x=i;sub(/.*/, "", x);print x,a[i]}}' $f4 > ${f4}_pci
if [ ! -s "${f4}_pci" ] ; then
>$f4_clone
fcinfo hba-portlnawk -F:' /OS Device/{print $2}' > ${f4}_wwns
while read -r wwn_cfg_dev
do
devcfg=`find_pci_addr "$wwn_cfg_dev"`
echo "$devcfg"|grep '0,' >/dev/null 2>&1
if [ $? -ne 0 ] ; then
nawk 'NR>4{if(/PCI){b=$1;sub("/.*"/, "", $1);a=$1;next}
{if($0~"$devcfg"){ar[a]=ar[a]?ar[a] FS $0:$0}
}}END{for(i in ar)print i,ar[i],("b")" >> "$f4_clone" ; for(i in ar)print i,ar[i]}' $f4 >> ${f4}_pci
fi
done < ${f4}_wwns
fi

```

```
}
```

```
function resetshll() {  
#echo -en "\033c" # reset term  
echo -en "\033[0m" # reset colours  
}
```

```
function screenw() {  
nawk -v c=$1 -v o=$2 'BEGIN{$ c =OFS=o;print}'  
}
```

```
function oldfiles() {  
if [ -s "$1" ] ; then  
newconff="no" ;  
else  
newconff="yes"  
fi
```

```
if [ ! -z "$3" ] ; then  
if [ ! -s "$1" ] ; then  
newconf="yes"  
else  
newconf="no"  
screenw 80 '#'  
oldconffiledetails=`ls -ltr $1nawk '{print $6,$7,$8,$NF}'`  
oldconffile=`echo "$oldconffiledetails"nawk '{print $NF}'`  
echo "Old inf files will be use -> $oldconffiledetails"  
screenw 80 '-'  
echo "If you want to Fresh [$3] RESULT(s) then Please REMOVE the $oldconffile "  
screenw 80 '#'  
echo  
((sleepstc++))  
if [ $sleepstc -eq 1 ] ; then  
sleepstc  
[ $? -eq 0 ] && sleepx="sleep 0.2" || sleepx="sleep 1"  
fi  
$sleepx  
fi
```

```
fi
}
```

```
function detectoldconf() {
if [ "$3" = "force" ] ; then
newconf="yes"
else
case $2 in
y) [ "$3" = "infunc" ] && newconf="yes" || oldfiles "$1" "$3" "$4" ;;
ynlnyn) oldfiles "$1" "$3" "$4" ;;
yy) newconf="yes" ;; # force YES
nn) newconf="no" ;; # force NO
*) oldfiles "$1" "$3" "$4" ;;
esac
fi
}
```

```
function testprtconf() {
if [ ! -x /usr/sbin/prtconf ] ; then
echo "prtconf cannot be run !! "
screenw 50 "-"
sleep 1
exit 1
fi
}
```

```
function testprtdiag() {
if [ ! -x /usr/sbin/prtdiag ] ; then
echo "prtdiag cannot be run !! "
screenw 50 "-"
sleep 1
exit 1
fi
}
```

```
function yncheck() {
checkit=`echo "$1"|nawk '{print $1}'`
echo "$checkit"|$grep -E "^y$|^n$|^y[y]$|^n[y]$" >/dev/null 2>&1
```

```

}

function check_params() {
if [ ! -z "$1" ]; then
echo "33[1;37m"
echo "cannot be detected any disk(s) or no disk info in the OLD cached information files .. ? "
fi

```

```

yncheck "$orgparams"
if [ $? -ne 0 ]; then
echo -e "You can try refresh the disk informations [ -> $0 y $orgparams ]n"
else
echo "$checkit" | $grep -E "^yy$" >/dev/null 2>&1
if [ $? -ne 0 ]; then
trychecks=`echo "$orgparams" | nawk '{for(i=2;i<NF;i++)printf "%s", $i FS; print $NF}`
echo "$checkit" | $grep -E "^y$" >/dev/null 2>&1
if [ $? -eq 0 ]; then
echo -e "You can try refresh the disk informations [ -> $0 yy $trychecks ]n"
else
echo -e "You can try refresh the disk informations [ -> $0 y $trychecks ]n"
fi
fi
resetshll
fi
}

```

```

function check_is_nodisk() {
if [ ! -z "$1" ]; then
$grep 'c.*t.*d' "$1" >/dev/null 2>&1
if [ $? -ne 0 ]; then
check_params "$?"
resetshll
exit 1
fi
else
check_params
fi
}

```

```

function is_show_or_print() {
echo -en "\033[1m"
if [ ! -s "$1" ]; then
echo
echo "Data file(s) are missing or cannot be found !! .."
check_is_nodisk
resetshll
exit 1
fi
[ $2 -eq 0 ] && cat "$1" | less -f -r "$1" ## -r parameter cause the bad screen scrolling
resetshll
}

```

```

function is_load_mpath() {
((countmpath++))
if [ $countmpath -eq 1 ]; then
modinfo -c >/dev/null 2>&1
if [ $? -eq 0 ]; then
modinfo -c |grep scsi_vhci >/dev/null 2>&1
if [ $? -ne 0 ]; then
echo "MPxIO modules cannot be found !! "
exit 1
else
modinfo -c |nawk '/scsi_vhci /'|grep -i UN
if [ $? -eq 0 ]; then
echo "MPxIO modules are NOT loaded !! "
exit 1
fi
fi
else
((testprtconf++))
[ $testprtconf -eq 1 ] && testprtconf
mpaths=`prtconf -vc /devices/scsi_vhci|nawk '/dev_link.*s2/{print;exit}'|$grep 'c.t.*'^`
if [ -z "$mpaths" ]; then
echo "MPxIO is seem INACTIVE !! "
exit 1;
fi
fi

```

```

fi
}

function disk_infos() {
((disk_infos++))
if [ $disk_infos -eq 1 ] ; then
detectoldconf "$f3" $cc "" "NEW DISK/LABEL infos from format"
if [ "$newconff" = "yes" ] ; then
echolformatlnawk '/^ *[0-9]./{print $2,$NF}'|$sed '/>/s/(.*) .*/1 No_LABEL/' > $f3 &
loopx "Preparing Format Disk Infos "
fi
fi
}

```

```

function format_disks() {
((format_disks++))
if [ $format_disks -eq 1 ] ; then
detectoldconf "$f9" $cc "" "NEW DISK device(s)"
if [ "$newconff" = "yes" ] ; then
disk_infos
# echo only disk infos from FORMAT
nawk '{print $1}' $f3 > $f9
fi
fi
}

```

```

function full_mpaths_slices() {
((full_mpaths_slices++))
if [ $full_mpaths_slices -eq 1 ] ; then
# disk infos with slices
detectoldconf "${f33}_slice" $cc "" "MPATHed DISKS Slices"
if [ "$newconff" = "yes" ] ; then
full_mpaths
$grep -E 'dsk' $f33 > ${f33}_slice
fi
fi
}

```

```

function istherelun() {
((istherelun++))
if [ $istherelun -eq 1 ] ; then
islun=`mpathadm list llnawk '/Path/{print ;exit}`
if [ -z "$islun" ] ; then
screenw 55 '#'
echo "No LUN(s) on the any initiator-port !! "
[ -z "$1" ] && exit 1
fi
fi
}

```

```

function full_mpaths() {
((full_mpaths++))
if [ $full_mpaths -eq 1 ] ; then
is_load_mpath
istherelun $1
if [ -z "$1" ] ; then
detectoldconf $f33 $cc "" "NEW MPATH Informations"
if [ "$newconff" = "yes" ] ; then
mpathadm list lu > $f33 &
loopx "Preparing Mpath Disk Infos "
fi
fi
fi
}

```

```

function mpaths_counts() {
# disk infos with paths
detectoldconf $f2 $cc "" "MPATHS Path Counts"
if [ "$newconff" = "yes" ] ; then
full_mpaths $1
$grep -E 'disk|Total Path' $f33|sed -e 's|/dev/[^ ]*/||' -e 's|.*/||' -e 's|Total Path Count:||'|nawk
'NR%2{printf "%s ",$0;next;}1' > $f2
fi
}

```

```

function mpath_disks() {

```

```

# echo only disk infos from MPATHADM
detectoldconf $f8 $cc "" "MPATH DISK DEVICES"
if [ "$newconff" = "yes" ] ; then
mpaths_counts $1
nawk '{print $1}' $f2 > $f8
fi
}

function istherehba() {
((istherehba++))
if [ $istherehba -eq 1 ] ; then
echo
ishba=`fcinfo hba-port | $grep HBA`
if [ -z "$ishba" ] ; then
screenw 55 '#'
fcinfo hba-port
exit 1
fi
fi
}

function hba_pci() {
istherehba
((hba_pci++))
if [ $hba_pci -eq 1 ] ; then
detectoldconf $f10 $cc "" "NEW HBA device(s)"
if [ "$newconff" = "yes" ] ; then
>$f10
>${f10}_clone
echo "Hba(s) information getting ..."
fcinfo hba-port | nawk '/HBA Port WWN:/{printf "%s ", $NF}/OS Device Name:/{printf "%s", $NF}/State:/{printf " %sn", $NF}' > ${f10}_temp
detect_pci_inf;
while read -r hba pci_cfg status; do
pci__address=`find_pci_addr "$pci_cfg`
fcinfo remote-port -s -p $hba 2>/dev/null | grep "OS Device" >/dev/null
[ $? -eq 0 ] && msg="[Disk_detected]" || msg="[Disk_not_detected]"
nawk '{for(i=2;i<=NF;i++){if($i=="$pci__address"){sub(".*@", "", $i); sub(/.*\/, "", $i); print

```

```

$1,["$i"]," "$hba", "$pci_cfg", "$status", "$msg" } }' ${f4}_pci >> $f10
nawk '{for(i=2;i<NF;i++){if($i=="$pci__address"){sub(".*@", "", $i);sub(/.*/, "", $i);print
$1,["$i"]," "$hba", "$pci_cfg", "$status", "$msg", $NF} } }' ${f4}_clone >> ${f10}_clone
done < ${f10}_temp &
loopx "HBA Port informations calculating "
echo -e "\n(HBA PCI Infos )\033[1m [Ready]\n"
fi
fi
}

function cal_dev_hba_port() {
detectoldconf "$f5" $cc "" "NEW Target PORT(s)"
if [ "$newconf" = "yes" ] ; then
check_luxadm
((cal_dev_hba_port++))
if [ $cal_dev_hba_port -eq 1 ] ; then
>$f5
full_mpaths_slices
# find only active hba(s)
while read -r disk ; do
luxadm display $disklnawk '/Controller/{a=$0}/Device Address/{b=$0}/WWN/{print $0 a b }'lnawk
'{print $5,$NF,$7}' > $f6
nawk '{n=split($NF,a,"/");printf "%s %s ",$2,$1;printf "%c","/";for(i=3;i<n-1;i++)printf
"%s",a[i]"/";printf "%c", "n"}' $f6 >>$f5
done < ${f33}_slice &
loopx "Preparing HBA Port Infos "
fi
fi
}

function isthereluxadm() {
isluxadm="luxadm -e port"
if [ -z "$isluxadm" ] ; then
screenw 55 '#'
echo "luxadm cannot get any FC device !! "
exit 1
fi
}

```

```

function check_luxadm() {
((check_luxadm++))
if [ $check_luxadm -eq 1 ] ; then
luxadm=/usr/sbin/luxadm #; [ -z "$luxadm" ] && luxadm=`which luxadm >/dev/null 2>&1`
if [ ! -x "$luxadm" ] ; then echo "luxadm cannot be found !! " ; exit 1 ; fi
isthereluxadm
fi
}

```

```

function cal_hba_ports() {
istherehba
# find only active hba ports
detectoldconf ${f6}_active $cc "" "Active HBA/PCI infos"
if [ "$newconf" = "yes" ] ; then
fcinfo hba-port|$grep -E 'HBA|State'|nawk '/State: online/{print x}{x=$NF}' >${f6}_active
fi

```

```

detectoldconf $f18 $cc "" "Inactive HBA ports"
if [ "$newconf" = "yes" ] ; then
fcinfo hba-port | nawk '/HBA/{print $4}' > $f17
# find only inactive hba port(s)
nawk 'NR==FNR{a[$1];next}{if(!($1 in a))print $1}' ${f6}_active $f17 > $f18
fi
}

```

```

function screento () {
printf "%sn%sn%sn%sn%sn" "Disk-Device Name [LUN]" "[VolName]" "HBA PCI Address" "-> "
"Storage UNIT" "FC HBA (Host) WWN(s)"
screenw 55 "--"
printf "%64s%sn%2sn%64s%sn%2sn%64s%sn%2sn" "====" " HBA Slots " "====" "====" " Remote WWN "
"====" "====" " Path State " "===="
screenw 55 "--"
}

```

```

function disk_hba_preap() {
detectoldconf $f7 $cc "$1" "NEW Target Ports with Cfg devices"
if [ "$newconf" = "yes" ] ; then
cal_dev_hba_port

```

```

hba_pci
nawk 'NR==FNR{a[$3]=$4;b[$3]=$1$2;next}{if($2 in a)print b[$2],$2,$1,$3,a[$2]}' $f10 $f5 > $f7
fi

```

```

detectoldconf $f25 $cc "$1" "NEW LABEL(s) for DISK(s)"
if [ "$newconf" = "yes" ] ; then
detectoldconf $f20 $cc "$1" "NEW PATHS"
[ "$newconf" = "yes" ] && disk_luxadm
disk_infos
nawk 'NR==FNR{a[$1]=$2;next}/c.*t.*d/{for(i in a)if($0~i)print $0,a[i]}!/c.*t.*d/{print}' $f3
${f20}_final > $f25 &
loopx "Label and Disk Paths informations calculating "
fi

```

```

detectoldconf $f27 $cc "$1" "PCI/Cfg/HBA/Target infos with SMART view"
if [ "$newconf" = "yes" ] ; then

```

```

detectoldconf $f26 $cc "$1" "PCI/Cfg/HBA/Target infos"
if [ "$newconf" = "yes" ] ; then
nawk 'NR==FNR{a[$2$3]=$1FS$NF;next}{if($3$2 in a)printf "%s %s %s %s
%s\n",a[$3$2],$1,$3,$2,$NF;else printf "%sn", $0}' $f7 $f25 > $f26 &
loopx "PCI/Cfg/Storage Path(s) "
fi

```

```

nawk '
/c.*t.*d/{diskmpath[$1]=$NF;size[$1]=$2FS$3;vendor[$1]=$4;d=$1}
!/c.*t.*d/{p=d$1FS$2FS$3;other[p]=other[p]RS$4FS$5FS$NF}
function lengthxxx(a,b,l){for(b in a)l++;return l}
function printfxx(c,ch){for(k=1;k<=c;k++)printf "%c",ch;printf "%c",RS}
END{
for(i in diskmpath){
print "Disk Boundary"
printfxx(110,"-")
printf "%-18s%34s%30s%28sn","MPATH LUN ID","Vendor","Size","Label_Name"
printfxx(110,"-")
ii=i
sub(".*dsk/", "",i);
printf "%-22s%20s%30s%20sn",i,vendor[ii],size[ii],diskmpath[ii];

```

```

printfxx(110,"=)
printf "%c",RS

for(j in other)
if(j~i){
split(j,arr,FS)
sub("^. *dsk/c.*t.*d.*s[0-9]+", "",arr[1])
xxx=lengthxxx(arr,b,l)
printfxx(80,"=)
printf "%-14s%4s%24sn", "PCI Address", "Path", "HBA Device"
printf "%-14s%4s%20sn", "HBA", "Channel", "Path"
printfxx(80,"=)

for(k=1;k<=xxx;k++)
printf "%-14s",arr[k]
printf "%c",RS
printfxx(80,"=)
printf "%50c%cn%50c%cn", " ", "|", " ", "|"

split(other[j],arrr,FS)
xxx=lengthxxx(arrr,b,l)
printf "%50c", " "
printfxx(51,"=)
printf "%58s%26s%17sn", "Host WWN", "Remote WWN", "Status"
printf "%50c", " "
printfxx(51,"=)

for(kk=1;kk<=xxx;kk+=3)
{
printf "%66s ",arrr[kk]
printf "%23s ",arrr[kk+1]
printf "%10s ",arrr[kk+2]
printf "%c",RS
}
printf "%c",RS
}
}
}' $f26 > $f27 &

```

```

loopx "Full Path Informations Preparing "
fi
}

function goto_disk_hba() {
for dsklbl in @$@
do
labeldiskcheck $dsklbl
if [ ! -z "$labelcheck" ] || [ ! -z "$diskcheck" ] ; then

detectoldconf ${f27}_$dsklbl $cc "" "FULL PATHS"
if [ "$newconf" = "yes" ] ; then
nawk '
/Disk Boundary/{writelabel($0,0)
while(getline){
if(!/Disk Boundary/){if(noprint==0)print $0}
else {ar="";writelabel($0,0);}}
function writelabel(a,i)
{
while(i++<3){getline;ar=ar?ar RS $0:$0}
{getline;if($NF=="$dsklbl"){print a RS ar RS $0;noprint=0}
else noprint=1 }
}' $f27 > ${f27}_$dsklbl
fi

is_show_or_print ${f27}_$dsklbl "$show"
else
nawk 'BEGIN{OFS=" ";print $0 RS " ==> Label or Disk cannot be found !! ['$dsklbl'] " RS $0 RS
}'
fi
done
}

function isgettingnewluns() {
nawk 'NR==FNR{a[$0];next}{if(!($0 in a))print}' ${1}.old $1|grep '! ' >/dev/null 2>&1
if [ $? -eq 0 ] ; then
checkfullpathlbl="yes"
else

```

```
checkfullpathlbl="no"
fi
}
```

```
function callcfgadm() {
cfgadm -o show_FCP_dev -al > $f12 &
loopx "Cfgadm calling "
cp $f12 ${f12}.old
[ $? -ne 0 ] && echo "Backup does not copied in the ${LOCAL_TEMP_DIR} "
}
```

```
function istherenewlun() {
if [ -s "$f12" ]; then
if [ -s "${f12}.old" ]; then
isgettingnewluns $f12
else
mv -f $f12 ${f12}.old
callcfgadm
isgettingnewluns $f12
fi
else
echo "new lun(s) test is fail !! "
echo "old informations cannot be found or the script may be running for the first time !! "
echo
checkfullpathlbl="no"
fi
}
```

```
function disk_hba() {
istherehba
is_load_mpath
istherelun
echo "$cc" | grep 'y' >/dev/null 2>&1
[ $? -eq 0 ] && istherenewlun
[ "$checkfullpathlbl" = "yes" ] && disk_hba_preap "infunc" || disk_hba_preap ""
```

```
echo -en "\33[1m"
[ ! -z "$1" ] && goto_disk_hba $@ || is_show_or_print $f27 "$show"
```



```
echo
[ $show -eq 0 ] && sleep 1 || sleep 2
fi
}
```

```
function detect_all_hba_pci() {
((detect_all_hba_pci++))
if [ $detect_all_hba_pci -eq 1 ] ; then
detectoldconf "$f41" $cc "" "HBA PCI DEVICE ADDRESS"
if [ "$newconf" = "yes" ] ; then
luxadm -e portlnawk '!/NOT/{print $1}' >$f41
fi
fi
}
```

```
function detect_all_pwwn() {
detect_all_hba_pci
detectoldconf "${f39}_pwwns" "$cc" "WWN TARGET PWWN(s)"
if [ "$newconf" = "yes" ] ; then
while read -r hbapci ; do
luxadm -e dump_map $hbapcilnawk 'NR>$1 {a[x++]= $4}END{for(i=0;i<x-1;i++)print a[i]}'
done <$f41lnawk '!a[$1]++' >${f39}_pwwns
fi
}
```

```
function check_pwwn() {
$grep "^${1}$" ${f39}_pwwns >/dev/null 2>&1
if [ $? -eq 0 ] ; then
no_pwwn=0
else
no_pwwn=1
echo "Bad Target WWN information -> [ $1 ] !! "
sleep 1
fi
}
```

```
function addlblspwwn() {
disk_infos
```

```
nawk 'NR==FNR{a[$1]=$2;next}{x=1;for(i in a)if($NF~i){x=0;print $0 FS FS FS a[i];next};if(x)print}'
$f3 $1 >${1}_labels &
loopx "Adding Disk Labels for [LUN] device(s) "
}
```

```
function getpwwninfo() {
((getpwwninfo++))
if [ $getpwwninfo -eq 1 ]; then
detectoldconf ${f39}_pwwns $cc "" "NEW TARGET PORT(s)"
if [ "$newconf" = "yes" ]; then
detect_all_pwwn
fi
fi
}
```

```
function getallluns() {
detectoldconf ${f39}_luns_all $cc "" "NEW LUNS from TARGET(s)"
if [ "$newconf" = "yes" ]; then
>${f39}_luns_all_temp
detect_all_hba_pci
while read -r hbapci ; do
luxadm -e dump_map $hbapci | nawk '{a=$4}END{print a}'; done <$f41
while read -r hba_wwn ; do
echo -e "HBA Port WWN: $hba_wwn"
fcinfo remote-port -s -p $hba_wwn 2>&1
echo "END"
done >>${f39}_luns_all_temp &
loopx "Getting all LUNs from Targets "
nawk '/HBA Port WWN:/{x=1}{if(/Remote Port/){if(x==1){print;x=0} else print RS $0} else print}'
${f39}_luns_all_temp > ${f39}_luns_all
fi
}
```

```
function getalllunslbl() {
detectoldconf ${f39}_luns_all_labels $cc "infunc" "NEW LUNS from All TARGETS with LABELS"
if [ "$newconf" = "yes" ]; then
getallluns
addlblspwwn ${f39}_luns_all
```

```

fi
}

function gettargetinfo() {
pwwnthis=$1
getalllunslbl
nawk 'BEGIN{$70=OFS="=";print}'
nawk '/HBA Port WWN:/{hba=$0;}/Remote Port WWN: '$pwwnthis'/{a[x++]=$0;while(getline){
if($0~/Remote Port WWN:\/!!NF){print hba;for(j=0;j<=x;j++)print a[j];x=0;break} else
{a[x++]=$0;}}}' ${f39}_luns_all_labels >>$2 &
loopx "... " "$!"
#loopx "Parsing the LUNS information for Target ['$pwwn'] " "$!"
echo
nawk 'BEGIN{$70=OFS="=";print "END for Target ['$pwwn'] " RS }' >>$2
}

```

```

function detect_pwwn_disk() {
check_luxadm
getpwwninfo

```

```

if [ -z "$1" ] ; then
allpwwns=""
else
allpwwns=$@
fi

```

```

if [ -z "$allpwwns" ] ; then
detectoldconf ${f39}_results_all $cc "" "NEW LUNS from Online TARGETS with LABELS"
if [ "$newconff" = "yes" ] ; then
>${f39}_results_all
while read -r pwwn
do
gettargetinfo $pwwn ${f39}_results_all
done <${f39}_pwwns
fi
is_show_or_print ${f39}_results_all "$show"

```

```

else

```

```

for pwwn in $allpwwns
do
check_pwwn $pwwn
>${f39}_results
if [ $no_pwwn -eq 0 ] ; then
gettargetinfo $pwwn ${f39}_results
is_show_or_print ${f39}_results "$show"
fi
done
fi
}

function detect_wwn_disk() {
check_luxadm
istherehba

if [ -z "$1" ] ; then
allwwns=`fcinfo hba-port|nawk '/HBA Port/{print $NF}`
else
allwwns=$@
fi

for i in $allwwns
do
hbawwn="$i"
check_hba $hbawwn
if [ $no_wwn -eq 0 ] ; then

if [ "$hba_state" = "online" ] || [ "$hba_state" = "ONLINE" ] ; then

detect_hba_cfg
pcicfg=`nawk ' $1=="$hbawwn" {print $2}' $f36`

detect_cfg_pci
pci_dev_hba=`nawk ' $1=="$pcicfg" {print $2}' $f37`

if [ ! -z "$pci_dev_hba" ] ; then
detectoldconf ${f37}_storage_results_$hbawwn $cc "" "HBA PORT CONFIGURATION"

```

```

if [ "$newconff" = "yes" ] ; then
nawk 'BEGIN{$80=OFS="="};print "Start of Storage Config Results for [" "'$hbawwn'" "]" RS $0 RS }'
>${f37}_storage_results_$hbawwn
nawk 'BEGIN{$80=OFS="="};print "Storage Port Configuration for ['$hbawwn']" RS "['$pci_dev_hba']"
RS $0 }' >>${f37}_storage_results_$hbawwn
luxadm -e dump_map "$pci_dev_hba" >>${f37}_storage_results_$hbawwn
nawk 'BEGIN{$80=OFS="="};print $0 RS RS "End of Results for [" "'$hbawwn'" "]" RS $0 RS }'
>>${f37}_storage_results_$hbawwn
fi
is_show_or_print ${f37}_storage_results_$hbawwn $show
fi

```

```

list_luns "$hbawwn"
[ $show -eq 1 ] && keycontinue list_luns
fi
fi
done
}

```

```

function get_dsk_infos() {
detectoldconf ${f37}_pwwn_dsk_${1} $cc "" "NEW LUN(s)"
if [ "$newconff" = "yes" ] ; then
fcinfo remote-port -s -p "$hbawwn" >${f37}_pwwn_dsk_${1} 2>&1
fi
}

```

```

function preapere_pwwn() {
hbawwn=$1
detectoldconf ${f37}_results_$hbawwn $cc "" "NEW LUN(s) with LABELS"
if [ "$newconff" = "yes" ] ; then
get_dsk_infos $hbawwn
((disk_infos_pwwn++))
[ $disk_infos_pwwn -eq 1 ] && disk_infos
nawk 'NR==FNR{if(/OS Device/)a[$NF];next}{for(i in a)if(i~$1)print i,$2}'
${f37}_pwwn_dsk_${hbawwn} $f3 >$f38
nawk 'BEGIN{$80=OFS="="};print "Start of LUNs Results for [" "'$hbawwn'" "]" RS $0 RS }'
>${f37}_results_$hbawwn
nawk 'BEGIN{$80=OFS="="};print "Storage Disk configuration for [" "'$hbawwn'" "]" RS

```

```

["$pci_dev_hba"] RS $0 }' >>${f37}_results_$hbawwn
nawk 'NR==FNR{a[$1]=$2;next}{if(/OS Device/){for(i in a)if(i~$NF)print $0,a[i]} else print}' $f38
${f37}_pwwn_dsk_$hbawwn >>${f37}_results_$hbawwn &
loopx "LUN(s) are getting from configurations "
nawk 'BEGIN{$80=OFS="="};print $0 RS RS "End of Results for [" "$hbawwn" "]" RS $0 RS }'
>>${f37}_results_$hbawwn
fi
}

```

```

function list_luns() {
preapere_pwwn $1
is_show_or_print ${f37}_results_$hbawwn "$show"
#less -f -r ${f37}_results_$hbawwn
echo
}

```

```

function dskcheck() {
dsklbl=$1
echo "$dsklbl"|$grep "/*dsk" >/dev/null 2>&1
[ $? -eq 0 ] && dsklbl=`echo "$dsklbl"|nawk -F '/' '{sub("s.*$", "", $NF);print $NF}`
}

```

```

function list_luns_disklbl() {
check_luxadm
getalllunslbl
if [ -z "$@" ]; then
is_show_or_print ${f39}_luns_all_labels "$show"
else
for dsklbl in $@
do
labeldiskcheck $dsklbl
if [ ! -z "$labelcheck" ] || [ ! -z "$diskcheck" ]; then
nawk -v dsklbl="$dsklbl" '
/HBA Port WWN:/{hba=$0;
while(getline)
{
if(/END/){break;}
if(/Remote Port WWN:/)

```

```

{
err=0;x=0
for(i=0;i<6;i++)
{
b[x++]=$0;
getline
if($0~"Error"){err=1;split("",b,RS);x=0;}
}
}
if(err==0)
if(!NF)split("",a,RS)
else
{
a[x++]=$0;
if($(NF-1)~dsklbl||$NF~dsklbl)
{
print "==== START =====" RS hba;
for(k=0;k<5;k++)print b[k]
for(i=x-4;i<x;i++)print a[i]
print "==== END =====" RS RS;
split("",a,RS);x=0;
}
}
}' ${f39}_luns_all_labels >${f41}_dsklbl

is_show_or_print ${f41}_dsklbl "$show"
else
nawk 'BEGIN{OFS=" ";print $0 RS " ==> Label or Disk cannot be found !! ['$dsklbl'] " RS $0 RS
}'
fi
done
fi
}

```

```

function hba_states() {
detectoldconf $f19 $cc "" "HBA STATE(s)"
if [ "$newconf" = "yes" ] ; then

```

```

screenw 55 '-' > $f19
nawk 'BEGIN{printf "%sn", "HBA ports status "}' >> $f19
screenw 55 '-' >> $f19
hba_pci
nawk 'BEGIN{l="-----"}{a[$1]=a[$1]RS$2FS$3FS$4FS$5;b[$1]=$NF}
END{for(i in a)printf "%s %sn%s%s",i,b[i],l,a[i] RS RS}' ${f10}_clone >> $f19
fi

```

```

echo -ne "\033[1m"
screenw 55 '-'
echo -n
cat $f19
screenw 55 '-'
}

```

```

function hba_info() {
istherehba
detectoldconf $f12 $cc "" "Cfgadm"
if [ "$newconf" = "yes" ]; then
callcfgadm
screenw 55 "#"
if [ ! -s $f12 ]; then
echo "Cfgadm cannot get the Fiber Controller ( HBA ) Ports !! "
exit 1
fi
fi
}

```

```

echo -e "System HBA Info "
screenw 65 '-'

```

```

total_fc_hba=`nawk '{if($2=="fc"||$2=="fc-fabric")x++}END{print x}' $f12`
echo -e "\033[0m ttt ${total_fc_hba} X FC_port \033[1m( Total)";
echo -en "E[35;40m"
nawk 'BEGIN{ORS=FS;printf "%25s",FS[" "]$2=="fc"||$2=="fc-fabric"&&$3=="connected"{printf "%s ",$1}END{printf "%s","]"RS}' $f12
echo
total_fc_hba_connect=`nawk '{if($2=="fc"||$2=="fc-fabric"&&$3=="connected")x++}END{print x}' $f12`

```

```

echo -e "\033[0m ttt ${total_fc_hba_connect} X FC_port \033[1m( Connected )";
echo -en "E[35;40m"
nawk 'BEGIN{ORS=FS;printf "%25s",FS[" "]$2=="fc"||$2=="fc-fabric"&&$3=="connected"{printf "%s ",$1}END{printf "%s","]"RS}' $f12
echo
total_fc_hba_inactive=\nawk '{if($2=="fc"||$2=="fc-fabric"&&$3=="connected"&&$4=="unconfigured")x++}END{print x}' $f12`
echo -e "\033[0m ttt ${total_fc_hba_inactive} X FC_port inactive \033[1m( Unconfigure ) ";
echo -en "E[35;40m"
nawk 'BEGIN{ORS=FS;printf "%25s",FS[" "]$2=="fc"||$2=="fc-fabric"&&$3=="connected"&&$4=="unconfigured"{printf "%s ",$1}END{printf "%s","]"RS}' $f12
echo
total_fc_hba_disconnect=\nawk '{if($2=="fc"||$2=="fc-fabric"&&$3=="disconnected")x++}END{print x}' $f12`
echo -e "\033[0m ttt ${total_fc_hba_disconnect} X FC_port \033[0mdisconnected \033[1m( Disconnected ) " ;
echo -en "E[35;40m"
nawk 'BEGIN{ORS=FS;printf "%25s",FS[" "]$2=="fc"||$2=="fc-fabric"&&$3=="disconnected"{printf "%s ",$1}END{printf "%s","]"RS}' $f12
echo
total_fc_hba_active=\nawk '{if($2=="fc-fabric"||$2=="fc"&&$3=="connected"&&$4=="configured")x++}END{print x}' $f12`
echo -e "\033[0m ttt ${total_fc_hba_active} X FC_port initialized state \033[1m( Configured ) " ;
echo -en "E[35;40m"
nawk 'BEGIN{ORS=FS;printf "%25s",FS[" "]$2=="fc-fabric"||$2=="fc"&&$3=="connected"&&$4=="configured"{printf "%s ",$1}END{printf "%s","]"RS}' $f12
sleep 1
echo -ne "E[0m"
echo -ne "\033[1m"
screenw 65 '-'
echo

```

```

detectoldconf ${f12}_cfgs_sorted $cc "" "Cfgadm HBA Infos "
if [ "$newconf" = "yes" ] ; then
fcinfo hba-portlnawk -F:' /OS Device/{x=$NF;sub(".*/", "",x)}/Manufac/{sub("^", "", $NF);xx=$NF}/Driver Name/{sub("^ ", "", $NF);printf "%8s%20s%8sn",x,("xx"),["$NF"]}'
>${f12}_cfgs

```

```

nawk 'BEGIN{$65=OFS="=";print "HBA Drivers" RS $0 }' >${f12}_cfgs_sorted

# sort minimum descending order @ygemici
nawk '{i=$1;sub(/^[^0-9]*/, "", $1);infos="";for(k=2;k<=NF;k++)infos=infos?infos FS $k:$k;a[x++]=$1
FS i FS infos;next}
function unsort(){for(j=0;j<x;j++){n=split(a[j],newa,FS);unsorted[j]=newa[1];;for(k=2;k<=n;k++)unsortedfull[j]=unsortedfull[j] FS newa[k]}}
function findminidx(minf,ll){for(j=0;j<ll;j++)if(minf==unsorted[j])return j}
function findmin(ll){for(j=0;j<ll;j++)min=(min<unsorted[j])?min:unsorted[j];return min}
function recal(arr){jj=0;for(j=0;j<x;j++)if(arr[j]){arr[jj]=arr[j];jj++;}}
END{unsort();ll=x;for(i=0;i<x;i++){min=unsorted[0];
newmin=findmin(ll);
minidx=findminidx(newmin,ll--);
n=split(unsortedfull[minidx],writex,FS);
printf "%26s ",writex[1];for(k=2;k<n;k++)printf "%4s ",writex[k];printf "%8sn",writex[n]
delete unsortedfull[minidx];
delete unsorted[minidx];;
recal(unsorted);recal(unsortedfull);
}}' ${f12}_cfgs >> ${f12}_cfgs_sorted

```

```

nawk 'BEGIN{$65=OFS="=";print}' >>${f12}_cfgs_sorted
fi

```

```

cat ${f12}_cfgs_sorted
}

```

```

function keycontinue() {
keycont="yes"
if [ ! -z "$1" ] ; then
echo -ne "ec"
keycont="no"
fi
echo -ne "\033[1m"
read -p "Press any key to continue ... [ or q - exit , m - return menu ] " X
if [ "$X" = "q" ] ; then
resetshll
exit
elif [ "$X" = "m" ] ; then

```

```

menu_0 1
fi
}

function fc_print() {
echo -n "\033[1m"
screenw 110 "#"
printf "%-18s %9s %18s %32s %26s n" "FC HBA (Host)" "Path" "Remote" "LUN:" "Label"
printf "%-18s %9s %18s %20s n" "Port WWN" "Channel" "WWN" " "
screenw 110 "#"
}

function echo_print() {
echo -n "E[0m"
echo -e "\nFC Port WWN E[37;41m$1\033[0m \033[1m -> ${3}.$4 <- \033[1m E[36;40m ( $2 )"
}

function hba_lun_path_msg() {
echo_print "$1" "$2" "$3" "$4"
fc_print
mpathadm list initiator-port | $grep $1 >/dev/null 2>&1
if [ $? -ne 0 ] ; then
echo "This HBA port does not configured for MPATH !! "
break
fi
}

function hba_pci_infos() {
remotewwn=$1
hostcfg=$2
hostwwn=$3

lunname="\luxadm display "$1" 2>/dev/null | $grep 'dev.*dsk' | $grep -i -v DEVICE"
if [ $? -ne 0 ] ; then
echo -e "\033[1m E[35;40m t t Any device cannot be found on the $j Remote Port !! \033[0m nn"
else
#echo -e "\033[1m E[35;40m Device Lists on the $j Remote Port \033[0m"
path_channel=`echo "$hostcfg"|nawk '{sub("/dev/cfg/", "", $1)}1`

```

c=1

DISKS * * (LUN) from the Storage Ports Configuration

```
for lunname_i in $lunname
```

```
do
```

```
#product=`luxadm display ${lunname_i} | grep Product | nawk -F: '{print $2}'`
```

```
#capacity=`luxadm display ${lunname_i} | grep capacity | nawk -F: '{print $2}'`
```

```
#echo -n "\033[0m" # green
```

```
echo -n "E[0m"
```

```
echo -n "\033[1m" # bold
```

```
nawk -v p="$path_channel" -v c=$c 'BEGIN{
```

```
if(c==1)printf "%-18s %9s %24s %54sn", "$hostwwn", p, "$remotewwn", "$lunname_i"
```

```
else printf "%108sn", "$lunname_i" }
```

```
#echo -ne "E[33;38m"
```

```
#printf "%70s %15s n" "${product}" "${capacity}"
```

```
c=$((c+1))
```

```
done
```

```
printf "%109sn" "END"
```

```
printf "%109snnn" "Total $((c-1)) LUN"
```

```
fi
```

```
}
```

```
function parse_hba_pci_infos() {
```

```
$sed 's//dev/.*/dsk///' $1 >${1}_sed
```

```
((disk_infos_countmpath++))
```

```
[ $disk_infos_countmpath -eq 1 ] && disk_infos
```

```
nawk 'NR==FNR{a[$1]=$2;next}{x=1;for(i in a)if($NF~i){x=0;print $0 FS FS FS a[i];next};if(x)print}'
```

```
$f3 ${1}_sed >${1}_labels &
```

```
loopx "Adding Disk Labels for [FC/Storage/Label] menu "
```

```
}
```

```
function hba_gets_lun_paths() {
```

```
detectoldconf ${f10}_wwn_results_${2}_labels $cc "" "New LUN(s) with Label "
```

```
if [ "$newconf" = "yes" ]; then
```

```
>${1}_results_$2
```

```
fcremote=`fcinfo remote-port -p "$2" | nawk '/Remote/{print $4}'`
```

```

for j in $fcremote
do
hba_lun_path_msg "$2" "$4" "$5" "$6" >>${1}_results_$2
hba_pci_infos "$j" "$3" "$2" >>${1}_results_$2 &
loopx "Host Lun Path information is getting for [$hba_port_wwn (WWN)] -> [$j (PWWN)] ..." "$!"
parse_hba_pci_infos ${1}_results_$2
done
fi
}

```

```

function scan_one_wwn() {
wwn=$1
state=`nawk '$3=="$wwn" {print $5}' $f10`
if [ "$state" = "offline" ] || [ "$state" = "OFFLINE" ]; then
configure="no"
else
configure="yes"
wwnline=`nawk '$3=="$wwn" {print $5}' $f10`
echo "$wwnline"|while read -r pci port hba_port_wwn cfg fcstate isdisk
do
hba_gets_lun_paths ${f10}_wwn "$i" "$cfg" "$fcstate" "$pci" "$port"
done
fi
}

```

```

#function detect_exits_wwn_all() {
# wwn Parser from full file
#if [ ! -s "${f10}_show_results_all ] ; then
#nawk '/$i/{a=$0;x++;if(x==0){getline b;getline c;getline d;getline e;a=a RS b RS c RS d RS
e}f=1;}FC Port WWN/{if(!$i/){f=0;exit}}f' ${f10}_show_results_all
#else
#no_wwn_all=1
#fi
#}

```

```

function detect_exits_wwn() {
wwnd=$1
scan_one_wwn "$wwnd"
}

```

```
if [ "$configure" = "yes" ] ; then
is_show_or_print "${f10}_wwn_results_${wwnd}_labels" "$show"
fi
}
```

```
function hba_lun_path() {
is_load_mpath
check_luxadm
```

```
wwnvals=$@
wwnvalsmodif=( $wwnvals )
```

```
if [ ! -z "$wwnvals" ] ; then
for i in $wwnvals
do
check_hba $i
if [ $no_wwn -eq 1 ] ; then
# delete the invalid wwn
wwnvalsmodif=( ${wwnvalsmodif[@]/$i} )
fi
done
if [ -z "$wwnvalsmodif" ] ; then
exit 1
else
```

```
case ${#wwnvalsmodif[@]} in
1) detect_exits_wwn $wwnvalsmodif
;;
*) for i in ${wwnvalsmodif[@]}
do
detect_exits_wwn "$i"
[ $show -eq 1 ] && keycontinue lun_path
done
;;
esac
fi
```

```
else
```

```

shownew=0
if [ $show -eq 0 ] ; then
detectoldconf ${f10}_show_results_all $cc "" "PCI-HBA-CFG device values"
if [ "$newconff" = "yes" ] ; then
shownew=1
else
isallwwninfile=`nawk '/FC Port WWN/{a[$4]}END{for(i in a)print i}' ${f10}_show_results_all`nawk
'END{print NR}`
allwwn=`nawk '{if($3)x++}END{print x}' $f10`
if [ $allwwn -eq $isallwwninfile ] ; then
cat ${f10}_show_results_all
else
shownew=1
fi
fi
fi

if [ $shownew -eq 1 ] ; then
>${f10}_show_results_all
fi

if [ $shownew -eq 1 ] || [ $show -eq 1 ] ; then

# always fresh status
while read -r -u 3 pci port hba_port_wwn cfg fcstate isdsk
do

if [ "$fcstate" = "online" ] || [ "$fcstate" = "ONLINE" ] ; then
# port online

detectoldconf ${f10}_wwn_results_${hba_port_wwn}_labels $cc "" "HBA/Storage Ports"
if [ "$newconff" = "yes" ] ; then
>${f10}_wwn_results_${hba_port_wwn}
hba_gets_lun_paths ${f10}_wwn "$hba_port_wwn" "$cfg" "$fcstate" "$pci" "$port"
else
if [ $show -eq 1 ] ; then
less -f -r ${f10}_wwn_results_${hba_port_wwn}_labels
keycontinue lun_path

```

```

fi
fi

if [ $shownew -eq 1 ] ; then
cat ${f10}_wwn_results_${hba_port_wwn}_labels >>${f10}_show_results_all
fi

elif [ "$fcstate" = "offline" ] || [ "$fcstate" = "OFFLINE" ] ; then
# port offline
[ $show -eq 1 ] && echo_print "$hba_port_wwn" $fcstate "$pci" "$port" || echo_print "$hba_port_wwn"
$fcstate "$pci" "$port" >>${f10}_show_results_all
echo

fi

done 3< $f10
exec 3>&-
fi
fi

if [ ! -z $shownew ] ; then
if [ $shownew -eq 1 ] ; then
cat ${f10}_show_results_all
fi
fi

if [ $show -eq 1 ] ; then
nawk 'BEGIN{$70=OFS="="};print RS RS $0 RS "END OF COMMAND" RS $0 RS }'
fi
}

function disk_format_preap() {
detectoldconf $f15 $cc "infunc" "DISK/LABEL infos from old saved files"
if [ "$newconff" = "yes" ] ; then
disk_infos
nawk 'BEGIN{$70=OFS="="};print "LABEL / DISK DEVICE(s)" RS $0 RS };{a[$2]++;b[$2]=b[$2] RS
$1}END{for(i in a)if(i in b)print i,"("a[i]" x disk
device)",RS"=====",b[i],RS}' $f3 > $f15 OFS=" "

```

```

fi
}

function goto_format_label() {
echo -en "\033[1m"
for formatlabel in $@
do
labelcheck "$formatlabel"
[ ! -z "$labelcheck" ] && nawk '/$formatlabel/{f=1;print RS $0 ;next}/^$/{f=0;}f;END{print}' $f15
2>/dev/null || echo "Label cannot be found [$formatlabel] !! "
#nawk '$1~/ $formatlabel/ && $1!~/c.*t.*d/{x=$0;getline;print x RS "=====" RS $0 RS}' $f15
2>/dev/null
done
}

```

```

function disk_format() {
disk_format_preap
if [ ! -z "$1" ] ; then
#nawk 'BEGIN{print "nSearching format disks";$70=OFS="=";print}'
echo -en "E[0m"
#echo -en "E[36;40mn"
goto_format_label $@
else
echo -e "E[0m"
echo -e "\033[1m"
check_is_nodisk "$f15"
is_show_or_print $f15 "$show"
fi
}

```

```

function detect_cfg_pci() {
detectoldconf $f37 $cc "" "PCI path(s)"
if [ "$newconff" = "yes" ] ; then
ls -ltr /dev/cfgnawk '/c[0-9]*/{sub("../.", "", $NF);sub(":.*", "", $NF);print $9, $NF}' >$f37
fi
}

```

```

function detect_hba_cfg() {

```

```

detectoldconf $f36 $cc "" "Cfg device(s)"
if [ "$newconff" = "yes" ] ; then
fcinfo hba-portlnawk -F/' /HBA Port WWN/{sub(".*: ", "");x=$0}/OS Device Name/{printf
"%s%10sn",x,$NF}' >$f36
fi
}

function check_offline_paths() {
nawk '/Initiator Port Name:/{initiator=$NF;getline;target=$NF;getline;getline;pathstate=$NF;if(pathstate
!="OK"){getline;printf "%10s%20s%13s%sn",target,initiator,pathstate,$0}}' $f43
}

function add_offline_paths() {
is_load_mpath
disk="$1"
mpathadm show lu $disk >$f43
check_offline_paths="`check_offline_paths`"
if [ ! -z "$check_offline_paths" ] ; then
check_offline_paths >${f43}_parse
detect_hba_cfg
nawk 'NR==FNR{a[$1]=$2;next}{if($2 in a)print $0,a[$2]}' $f36 ${f43}_parse >${f43}_parse_edit
detect_cfg_pci
nawk 'NR==FNR{a[$1]=$2;next}{if($NF in a)printf
"%-54s%24s%22s%11s%14s%3sn",a[$NF],$1,$2,$3,$4,$5,$6}END{print RS}' $f37 ${f43}_parse_edit
>>$f20
fi
}

# maybe for later improves ( about Flash Card ( F40 / F80 .. ) and Local ( non-SAN ) Scsi/SSD disk
'Size' and 'Vendor' informations ....
#function add_other_disks() {
#echolformatlsed '/>/s/(.*) .*/1 No_LABEL/' >${f3}_full
#nawk -F/' '{sub("s[0-9]*$", "", $NF);print $NF}'
#size=`grep $1 $f20|nawk '{print $2}' 2>/dev/null `
#if [ -z "$size" ] ; then
#vendor=`grep $1 $f20|nawk '{print $3}' 2>/dev/null `
#if [ -z "$vendor" ] ; then
#.....

```

```
#}
```

```
function disk_luxadm() {
check_luxadm
((disk_luxadm++))
if [ $disk_luxadm -eq 1 ] ; then
>$f20
full_mpaths_slices
while read -r disk ; do
luxadm display "$disk"lnawk -v dsk="$disk" 'BEGIN{printf
"%s",dsk}{if(/Vendor/)a=$NF;if(/Product/){v=a"_"$NF};if(/capacity/)printf "%44s%s
%22sn",$(NF-1)," MB",v;if(/Controller/)printf "%sn",$(NF);if(/Device Address//WWN//State/)printf
"%sn",$(NF)}lnawk 'NR==1 {print;}NR!=1 {if(NR%4==1){printf "%10sn",$(x="")}else{if(!x){printf
"%-60s",$(x=1)}else printf "%20s",$(x)}}' >>$f20
add_offline_paths "$disk"
done < ${f33}_slice &
loopx "Disk and Port Informations Preparing "
```

```
# leave scsi controller and other disks for now -> "add_other_disks"
no_san=`nawk '/c.*t.*d/{if($2=="")a=a?a FS NR:NR}END{print a}' $f20`
nawk -v nowrite="$no_san" 'BEGIN{m=split(nowrite,nosan,FS);} {now=0;for(i=1;i<=m;i++)if(NR!=nos
an[i])now++;if(now==m)print}' $f20 >${f20}_final
fi
}
```

```
function goto_mpaths_label_capt() {
echo -n "E[33;36mn"
nawk 'BEGIN{printf "n%20sn","SAN disks with MPATHed *MPxIO paths "'
screenw 80 "#"
echo -n "E[0m"
screenw 111 "#"
nawk 'BEGIN{printf "%s %30s %16s %18s %17sn","Disk/Vendor_Product/Size","Label
Name","WWPN,LUN","WWN","STATE"}'
screenw 111 "#"
echo -n "E[0m"
echo -n "E[33[1m"
}
```

```

function disk_mpaths_preap() {
detectoldconf $f20 $cc "infunc" "NEW PATHS"
[ "$newconff" = "yes" ] && disk_luxadm
cal_dsk_lbl_path
nawk 'BEGIN{l="====="}NR==FNR{a[$1]=$0
;next}/c.*t.*d/{for(i in a)if($1~i){f++;if(f==1)print l RS a[i] RS "["$3]" RS ("2FS$3)" RS l;else
print RS l RS a[i] RS ["$4]" RS ("2FS$3)" RS l;next}}/c.*t.*d/{f=0;}f' $f1 ${f20}_final >>$f21 &
loopx "Disk HBA Ports informations calculating "
}

```

```

function goto_mpaths_label() {
for pathlabel in $@
do
((c++))
detectoldconf ${f21}_${pathlabel} $cc "infunc" "DISK PATH(s)"
if [ "$newconff" = "yes" ] ; then
[ $c -eq 1 ] && goto_mpaths_label_capt > ${f21}_${pathlabel}
nawk '{if($2~/${pathlabel}/||$1~/${pathlabel}/){print x RS $0;while(getline){if(NF)print $0;else {print
RS;break}} }else {x=$0}}' $f21 >>${f21}_${pathlabel}
fi
is_show_or_print ${f21}_${pathlabel} "$show"
done
}

```

```

function check_full_pathlabel() {
detectoldconf $f21 $cc "$1" "PATHS with LABELS"
grep 'c.*t.*d' $f21 >/dev/null 2>&1
if [ $? -ne 0 ] || [ "$newconff" = "yes" ] ; then
screenw 80 '#'
goto_mpaths_label_capt > $f21
disk_mpaths_preap
fi
}

```

```

function cal_dsk_lbl_path() {
((count_cal_dsk_lbl_path++))
if [ $count_cal_dsk_lbl_path -eq 1 ] ; then
detectoldconf $f1 $cc "" "DISK/LABEL/PATH infos"

```

```

if [ "$newconff" = "yes" ] ; then
disk_infos
mpaths_counts
nawk 'NR==FNR{a[$1]=$2;next}{if($1 in a)printf "%-48s%17s%24s%sn",$1,$2,a[$1]," x Path"}' $f2
$f3 >$f1
fi
fi
}

```

```

function labelcheck() {
disk_infos
label=$1
labelcheck=`nawk '{a[$2]}END{if("$label" in a)print "OK"}' $f3`
# [ -z "$labelcheck" ] && echo -e "Label cannot be found [$label] !! "
}

```

```

function diskcheck() {
disk_infos
disk=$1
diskcheck=`nawk -v disk="$disk" 'disk~$1{print "OK"}' $f3`
# [ -z "$diskcheck" ] && echo -e "Disk cannot be found [$disk] !! "
}

```

```

function disk_mpaths() {
istherehba
is_load_mpath
istherelun
echo "$cc" | grep 'y' >/dev/null 2>&1
[ $? -eq 0 ] && istherenewlun
[ "$checkfullpathlbl" = "yes" ] && check_full_pathlabel "infunc" || check_full_pathlabel ""
echo -e "\033[0m"
if [ ! -z "$1" ] ; then
lblskvals=( $@ )
for dsklbl in ${lblskvals[@]}
do
labelcheck $dsklbl
if [ -z "$labelcheck" ] ; then
fdsklbl=$dsklbl

```

```

dskcheck $dsklbl
ndsklbl=$dsklbl
if [ "$ndsklbl" != "$fdsklbl" ] ; then
lbdskvals=( ${lbdskvals[@]/$fdsklbl} )
lbdskvals=( ${lbdskvals[@]} $ndsklbl )
fi
diskcheck $dsklbl
fi
if [ -z "$diskcheck" ] && [ -z "$labelcheck" ] ; then
nawk 'BEGIN{$80=OFS="=";print $0 RS " ==> Label or Disk cannot be found !! ['$dsklbl'] " RS $0 RS
}'
lbdskvals=( ${lbdskvals[@]/$dsklbl} )
fi
done
if [ ! -z "$lbdskvals" ] ; then
goto_mpaths_label ${lbdskvals[@]}
fi
else
check_is_nodisk "$f21"
is_show_or_print $f21 "$show"
fi
}

```

```

function non_mpaths() {
screenw 80 "#"
format_disks
mpath_disks
# if no MPATH disks
if [ ! -s "$f8" ] ; then
check_is_nodisk "$f9"
is_show_or_print $f9 "$show"
else
nawk 'NR==FNR{a[$1];next}{if(!($1 in a))print $1}' $f9 $f8 > $f14
if [ -s "$f14" ] ; then
nawk 'BEGIN{$80=OFS="=";print RS "SAN/SCSI disks without MPxIO " RS $0 }'
nawk 'NR==FNR{a[$1];next}{if($1 in a)print $1,$2}' $f14 $f3
else
echo -ne "E[36;40m"

```

```
echo "No Disk(s) Found non-MPxIO !!"
```

```
echo
```

```
fi
```

```
fi
```

```
}
```

```
function disk_errors_capt() {
```

```
  echo `nawk '{printf "nn%s%29sn%46sn", "[Disk-Device
```

```
Name]", "[VolName]", "[Errors]"}END{ $80=OFS="=";printf "%s%s", $0,RS;}'
```

```
}
```

```
function disk_errors_preap() {
```

```
  disk_errors_capt > $f22
```

```
  disk_infos
```

```
  nawk 'NR==FNR{a[$1]=$2;next}{for(i in a)if($1==i)printf "%20s %sn%42s %s %s %4s %s %s %4s %s %sn", $1,a[$1],$2,$3,$4,$5,$6,$7,$8,$9,$10}' $f3 $f13 >> $f22
```

```
}
```

```
function disk_sorted_errors() {
```

```
  disk_errors_capt > ${1}_sorted
```

```
# sort maximum ascending order @ygemici
```

```
nawk '/c.*t.*d/{ disk=$1;label=$2;getline;if(!/. *Errors: 0.*Errors: 0.*Errors:
```

```
0/){a[disk]=$0;b[disk]=label;gsub(/ [^ ]* Errors: /,FS);;
```

```
for(i=1;i<=NF;i++)newmax=maxf($i,newmax);maxar[disk]=newmax;;newmax=0}}
```

```
function maxf(a,max){ max=(max>a)?max:a;return max }
```

```
function sorted(array,arraymax){ for(i in
```

```
array){ arraymax2=arraymax;arraymax=maxf(array[i],arraymax);
```

```
if(arraymax>arraymax2)j=i}{ print j,b[j]"n"a[j];delete maxar[j]}}
```

```
function isEmpty(arr,idx){ for(idx in arr)return 0;return 1 }
```

```
END{ while(!isEmpty(maxar))sorted(maxar,newarraymax);}' $1 >> ${1}_sorted &
```

```
loopx "Errors are getting and sorting "
```

```
echo -ne "␣33[1m"
```

```
is_show_or_print "${1}_sorted" "$show"
```

```
}
```

```
function goto_diskerrors_label() {
```

```
  [ ! -s "$f22" ] && disk_errors_preap
```

```

disk_sorted_errors $f22
disk_errors_capt > ${f22}_sorted_full
echo -ne "\033[1m"
for errorlabel in $@
do
nawk '/$errorlabel/{x=$0;getline;print x RS $0 }' ${f22}_sorted >> ${f22}_sorted_full
done
is_show_or_print "${f22}_sorted_full" "$show"
}

```

```

function disk_errors() {
iostat -En > $f13
grep '.*Errors: .*Errors: .*Errors: ' $f13|grep -v '.*Errors: 0.*Errors: 0.*Errors: 0' >/dev/null 2>&1
if [ $? -ne 0 ] ; then
nawk 'BEGIN{$80=OFS="="};print $0 RS " ==> No Error(s) found .. " RS $0 }'
exit 1
fi
disk_errors_preap;
[ ! -z "$1" ] && goto_diskerrors_label "$@" || disk_sorted_errors $f22
}

```

```

function labeldiskcheck() {
dsklbl=$1
labelcheck $dsklbl
if [ -z "$labelcheck" ] ; then
dskcheck $dsklbl
diskcheck $dsklbl
fi
}

```

```

function inq_ok() {
if [ -x "$1" ] ; then
inqexec="$1"
else
echo -e "-> $1 <- executable cannot be found in the '$LOCAL_WORK_DIR' directory !! n"
exit 1
fi
}

```

```

function inq_chk(){
((inq_chk++))
if [ $inq_chk -eq 1 ] ; then
systeminfo=`isainfo -b 2>/dev/null`
case "$systeminfo" in
32)echo "32 bit system does not supported !! " ; exit 1
;;
64)isainfo|$grep sparc >/dev/null 2>&1 ; [ $? -eq 0 ] && arch="sparc_64" || arch="x86_64"
;;
*)echo "System architecture cannot be found !! " ; exit 1
;;
esac

```

```

LOCAL_WORK_DIR="/usr/local/bin"

```

```

case "$arch" in
sparc_64)inq_ok "$LOCAL_WORK_DIR/inq.sol64"
;;
x86_64)inq_ok "$LOCAL_WORK_DIR/inq.solarisx86_64"
;;
esac
fi
}

```

```

function inq_sym_output_label_capt() {
nawk '
NR==7||NR==9{printf "%s%s\n", $0, "-----"}
NR==8{printf "%s%26sn", $0, "Label Name"}
NR>9{printf "%sn", $0}
'
}

```

```

function inq_vol() {
detectoldconf $3 $cc "" "NEW Inquiry Disk(s) with Label(s) for $4"
if [ "$newconff" = "yes" ] ; then
vendor="$4"
production="$5"
nawk 'BEGIN{print $0 RS "'$vendor' '$production'"

```

```

}NR==FNR{a[$1]=$2;next}{if(FNR>0&&FNR<4)print;for(i in a){if($1~i)printf "%s%12sn", $0,a[i]}}
$1 $2 > $3 &
loopx "inquiry and label informations calculating "
fi
}

```

```

function inq_sym_wnw_preap() {
detectoldconf ${f28}_vendors $cc "infunc" "NEW VENDOR(s) informations"
[ "$newconff" = "yes" ] && $inqexec -no_dotslnawk 'NR>9{sub(":", "", $2);sub(":", "", $3);a[$2 FS
$3]}END{for(i in a)print i}' >${f28}_vendors
$inqexec -hl$grep _wnw > ${f29}_storage_lst
$grep -i Hitachi ${f28}_vendors >/dev/null 2>&1
[ $? -eq 0 ] && echo "-hds_wnw : display HITACHI WWN" >> ${f29}_storage_lst

```

```

while read -r san_vendor san_prod
do
isvendor=`$grep -i "$san_vendor" ${f29}_storage_lst`
[ ! -z "$isvendor" ] && vendor_match="`echo $isvendor|nawk '{print $1}'`"
isprod=`$grep -i "$san_prod" ${f29}_storage_lst`
[ ! -z "$isprod" ] && vendor_match="`echo $isprod|nawk '{print $1}'`"
if [ ! -z "$vendor_match" ] ; then
$inqexec "$vendor_match" -no_dots | inq_sym_output_label_capt > $f28
disk_infos
inq_vol $f3 $f28 ${f28}_vendor_${san_vendor} ${san_vendor} ${san_prod}
vendor_match=""
fi
done < ${f28}_vendors
}

```

```

function goto_inq_sym_wnw() {
#detectoldconf $f29 $cc "" "NEW Inquiry Disk(s) with Label(s)"
#[ "$newconff" = "yes" ] && $inqexec -no_dotslnawk 'NR>4' >$f29
for dsklbl in $@
do
labeldiskcheck $dsklbl
if [ ! -z "$labelcheck" ] || [ ! -z "$diskcheck" ] ; then
>"${f29}_${dsklbl}_s"
for vendorlst in ${f28}_vendor_*

```

```

do
$grep " $dsklbl$ " $vendorlst >/dev/null 2>&1
x_x=$?
if [ "$x_x" -eq 0 ] ; then
nawk -v lbl=$dsklbl 'NR<=5||$NF==lbl' $vendorlst >>"${f29}_${dsklbl}_s"
nawk 'BEGIN{$120=OFS="="};print $0 RS }' >> "${f29}_${dsklbl}_s"
fi
done
is_show_or_print "${f29}_${dsklbl}_s" "$show"
else
nawk 'BEGIN{$80=OFS="="};print $0 RS " ==> Label or Disk cannot be found !! ['$dsklbl'] " RS $0 RS
}'
fi
done
}

```

```

function inq_sym_wnn() {
istherehba
inq_chk;
inq_sym_wnn_preap
echo -e "␣33[1m"
[ ! -z "$1" ] && goto_inq_sym_wnn $@ || for vendor_disks in ${f28}_vendor_* ; do is_show_or_print
"$vendor_disks" "$show" ; done
}

```

```

function goto_inq_hba() {
for hba in $@
do
check_hba $hba
if [ $no_wnn -eq 0 ] ; then
upper_wnn=`echo $hba|sed
'y/abcdefghijklmnopqrstuvwxyZ/ABCDEFGHIJKLMNopqrstuvwxyz/'`
nawk 'BEGIN{$70=OFS="="};print "inq HBA informations " RS $0}
{if(NR>6){
a[x++]=0;if(/port WWN/){if($NF~"$upper_wnn"){w=1;}}
if(/^\$/||/-----/){if(w==1){for(i=0;i<=x;i++)print a[i];exit}else split("",a,RS);x=0;}
}
}' $f32

```

```

fi
done
}

function inq_hba(){
istherehba
inq_chk;
[ ! -s "$f32" ] && $inqexec -hba > $f32
echo -e "\33[1m"
[ -z "$1" ] && is_show_or_print $f32 "$show" || goto_inq_hba $@
}

function goto_inq_vendor() {
nawk 'NR>0&&NR<6' $f31 > ${f31}_${dsklbl}
for dsklbl in $@
do
labeldiskcheck $dsklbl
if [ ! -z "$labelcheck" ] || [ ! -z "$diskcheck" ]; then
nawk -v dsklbl="$dsklbl" '{if($0~dsklbl)print}' $f31 >> ${f31}_${dsklbl}
is_show_or_print ${f31}_${dsklbl} "$show"
else
nawk 'BEGIN{$80=OFS="="};print $0 RS " ==> Label or Disk cannot be found !! ['$dsklbl'] " RS $0 RS
}'
fi
done
}

function inq_vendor_output_label_capt() {
nawk '
NR==7||NR==9{printf "%s%s\n", $0, "-----"}
NR==8{printf "%s%12sn", $0, "Label Name"}
NR>9{printf "%sn", $0}
'
}

function inq_vendor_preap() {
$inqexec -no_dots | inq_vendor_output_label_capt > $f30
disk_infos
}

```

```
inq_vol $f3 $f30 $f31
}
```

```
function inq_vendor() {
inq_chk;
detectoldconf $f31 $cc "" "NEW Inquiry Disk Vendor(s) with Label(s)"
if [ "$newconff" = "yes" ]; then
inq_vendor_preap
fi
echo -e "\033[1m"
[ ! -z "$1" ] && goto_inq_vendor $@ || is_show_or_print $f31 "$show"
}
```

```
##### messages error finder ; its just hobby :) 19/10/16
##### @ygemici unix.com #####
```

```
##### MESSG DATES #####
```

```
function perlcheck() {

if perl < /dev/null > /dev/null 2>&1 ; then
modules=Time::Local
perl -M"$modules" -e "print "Module installed.\n";" >/dev/null 2>&1
if [ $? -ne 0 ] ; then echo " 'Time::Local' module cannot be found " ; exit 1 ;fi
else
echo " -> Perl not found , dates does not works "
exit 1
fi
}
```

```
function find_global_fs() {
global_fs=`printf "%sn" "$@"|sed 's/[^ ,;]//g'`
}
```

```
function find_date_fs() {
r_fs=`printf "%sn" "$@"|sed 's/[0-9A-Za-z]*/ /g'|nawk '{for(i=1;i<=NF;i++)print $i}'`
if [ -z "$r_fs" ] ; then
printf "%sn" "No separator or column [ dd,mm ] -> $@ !! "
```

```
dateFormat_error "Syntax"
fi
```

```
mn_r_fs=`echo "$r_fs"|nawk '!a[$1]++'|nawk 'END{print NR}`
if [ $mn_r_fs -gt 1 ]; then
printf "%sn" "Multiple separator or column [ dd,mm ] -> $@ !! "
dateFormat_error "Syntax"
fi
```

```
n_r_fs=`echo "$r_fs"|nawk 'END{print NR}`
if [ $n_r_fs -gt 1 ]; then
printf "%sn" "Extra separator or column [ dd,mm ] -> $@ !! "
dateFormat_error "Syntax"
fi
```

```
fs=`echo "$r_fs"|nawk '!a[$1]++`
}
```

```
perl_month_convert() {
echo "$1"|$grep -i -E "^Jan$|^Feb$|^Mar$|^Apr$|^May$|^Jun$|^Jul$|^Aug$|^Sep$|^Oct$|^Nov$|^Dec$"
>/dev/null 2>&1
if [ $? -eq 0 ]; then
mon=`month_convert "$1"`
perl_convert_month=`nawk -v mon="$mon" 'BEGIN{split("Jan Feb Mar Apr May Jun Jul Aug Sep Oct
Nov Dec",month," ");for(i=1;i<=12;i++)if(month[i]~mon)print i-1}`
else
echo $1|$grep -E "^0[1-9]$|^1[0-2]$" >/dev/null 2>&1
if [ $? -eq 0 ]; then
perl_convert_month=`nawk -v mon=$1 'BEGIN{print mon-1}`
fi
fi
[ -z "$perl_convert_month" ] && dateFormat_error "$1"
}
```

```
function datecal() {
perlcheck;
perlday=$1
perlmonth=$2
```

```
perlyear=`date +%Y`
```

```
[ -z "$perlday" ] && dateFormat_error $perlday
```

```
[ -z "$perlmonth" ] && dateFormat_error $perlmonth
```

```
[ -z "$perlyear" ] && dateFormat_error $perlyear
```

```
epochgiven=$(perl -e 'use Time::Local; print timelocal(0,0,0,$perlday,$perlmonth,$perlyear)')
```

```
[ $? -ne 0 ] && dateFormat_error "$perlday/$((perlmonth+1))/$perlyear"
```

```
[ $nextday -eq 1 ] && daysecs=86400 || daysecs=0 ;
```

```
day_month=`perl -e 'print scalar(localtime('$epochgiven'+'$daysecs'))'|nawk '{print $3,$2,$NF;exit}'`
```

```
newday=`echo "$day_month"|nawk '{print $1}'`
```

```
newmonth=`echo "$day_month"|nawk '{print $2}'`
```

```
newyear=`echo "$day_month"|nawk '{print $3}'`
```

```
}
```

```
function nextday() {
```

```
perlcheck;
```

```
perlday=$1
```

```
perlyear=`date +%Y`
```

```
month=`date +%m`
```

```
perlmonth=`nawk 'BEGIN{print '$month'-1}'`
```

```
epochgiven=$(perl -e 'use Time::Local; print timelocal(0,0,0,$perlday,$perlmonth,$perlyear)')
```

```
nday=$(perl -e 'print scalar(localtime('$epochgiven'+86400))'|nawk '{print $3}')
```

```
}
```

```
function dateFormat_error() {
```

```
if [ "$1" = "Syntax" ]; then
```

```
echo -e "Usage -> '[day][FS][month]233[1m[ ,;]233[0m[day][FS][month]' "
```

```
else
```

```
echo "Some problems in the date formats -> $@"
```

```
fi
```

```
exit 1
```

```
}
```

```
function preapdates() {
```

```
x=$1
```

```

for i in day month year nextday
do
eval echo "${i}$x"
done
day="";month="";year=""
}

```

```

function process() {
x=${@: -1}

```

```

[ -z $1 ] && eval day$x="NULL" || eval nextday$x=1
[ -z $2 ] && eval month$x="NULL"
[ -z $3 ] && eval year$x="NULL"

```

```

preapdates $x
}

```

```

function day_check() {
echo "$1" | $grep -E "^0[1-9]$" >/dev/null 2>&1
[ $? -eq 0 ] && daycheck=`nawk -v day=$1 'BEGIN{printf "%d",day}`

```

```

echo "$1" | $grep -E "^0[1-9]$" | ^[1-9] | ^[1-2][0-9] | ^3[0-1]$" >/dev/null 2>&1
[ $? -eq 0 ] && daycheck="" || dateFormat_error $1
}

```

```

function month_convert() {
echo $1 | $grep -E "^0[1-9]$" | ^[1-9] | ^1[0-2]$" >/dev/null 2>&1
if [ $? -eq 0 ] ; then
nawk -v mon="$1" 'BEGIN{split("Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec",month,"");for(i=1;i<=12;i++)if(i==mon)print month[i]}'
fi

```

```

echo "$1" | $grep -E "^Jan$" | ^Feb$" | ^Mar$" | ^Apr$" | ^May$" | ^Jun$" | ^Jul$" | ^Aug$" | ^Sep$" | ^Oct$" | ^Nov$" | ^Dec$"
>/dev/null 2>&1
if [ $? -ne 0 ] ; then
for i in Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
do
echo $1 | $grep -i $i > /dev/null

```

```

[ $? -eq 0 ] && echo "$i"
done
else
echo $1
fi
}

function month_check() {
x=$2

eval err$x=0
echo $1|$grep -E "^0[1-9]${1}^[1-9]${1}^[0-2]${1}" >/dev/null 2>&1
[ $? -eq 0 ] && eval month${x}=`month_convert $1` || eval err$x=1

check_error=`eval echo '$err$x`
if [ $check_error -eq 1 ] ; then
echo "$1"|$grep -i -E "^Jan${1}^Feb${1}^Mar${1}^Apr${1}^May${1}^Jun${1}^Jul${1}^Aug${1}^Sep${1}^Oct${1}^Nov${1}^Dec${1}"
>/dev/null 2>&1
[ $? -eq 0 ] && eval month${x}=`month_convert $1` || eval err$x=2
fi

check_error=`eval echo '$err$x`
[ $check_error -eq 2 ] && dateFormat_error $1
}

function start_check() {
x=$3
if [ "$1" = "NULL" ] ; then
eval day$x=""
else
day_check $1
[ ! -z $daycheck ] && eval day$x=$daycheck
fi

if [ "$2" = "NULL" ] ; then
monthdate=`date +%m`
eval month$x=`month_convert $monthdate`
else

```

```
month_check $2 $x
```

```
fi
```

```
# year implementation cancelled
```

```
#[ $4 = "NULL" ] && eval year$x=`date +%Y`
```

```
}
```

```
function date1_parser() {
```

```
start1=0
```

```
nextday1=0
```

```
dates1=( `echo "$@" | while IFS="$fs" read -r day1 month1 ; do
```

```
process "$day1" "$month1" 1 ;
```

```
done ` )
```

```
for i in day month
```

```
do
```

```
eval ${i}1=${dates1[$xx]}
```

```
((++xx))
```

```
done
```

```
start_check "$day1" "$month1" 1
```

```
}
```

```
function date2_parser() {
```

```
start2=0
```

```
nextday2=0
```

```
dates2=( `echo "$@" | while IFS="$fs" read day2 month2 ; do
```

```
process "$day2" "$month2" 2
```

```
done ` )
```

```
for i in day month nextday
```

```
do
```

```
eval ${i}2=${dates2[$xxx]}
```

```
((++xxx))
```

```
done
```

```
start_check $day2 $month2 2
```

```

if [ $nextday2 -eq 1 ] ; then
nextday=$nextday2
perl_month_convert $month2
datecal $day2 $perl_convert_month
fi
}

```

```

function find_nextmonth() {
nawk -v monthsearch="$1" 'BEGIN{split("Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov
Dec",month," ");for(i=1;i<=12;i++)if(monthsearch~"Dec"){print "Jan";exit} else
{if(month[i]~monthsearch)print month[i+1]}}'
}

```

```

function dmesg_date_parser() {
echo '$@'|grep '_x_' >/dev/null 2>&1
[ $? -eq 0 ] && exit 1 # it is a bug ?
nrdates=`printf "%sn" "$@"|$sed 's/[ ,;]/_x_/g'|nawk -F'_x_' '{print NF}'`

```

```

case $nrdates in
1)find_date_fs "$@"
date1_parser "$@"
[ -z "$day1" ] && day1="empty"
call_results "$day1" "$month1" 1 #"$newday" "$newmonth"
###echo FIRST DATE "$month1 $day1"
###echo LAST DATE "$newmonth $newday" #newyear twodates
;;

```

```

2) find_global_fs "$@" ;
printf "%sn" "$@"|while IFS="$global_fs" read date1 date2 ; do
find_date_fs "$date1"
date1_parser "$date1"
orgday=$day1
orgmonth=$month1
###echo FIRST DATE "$month1 $day1"
[ -z "$day1" ] && day1="empty"

```

```

find_date_fs "$date2"
date2_parser "$date2"

```

```
orgfday=$day2
orgfmonth=$month2
```

```
[ -z "$day2" ] && day2="empty"
```

```
call_results "$day1" "$month1" 2
call_results "$day2" "$month2" 3
###echo SECOND LAST DATE "$newmonth $newday"
```

```
done
```

```
::
```

```
*)dateFormat_error "Syntax"
```

```
::
```

```
esac
```

```
}
```

```
function find_last_day() {
perl_month_convert $1
month=$perl_convert_month
year=`date +%Y`
perl -e 'use strict;use Time::Local;my $time = timelocal(0,0,0,1,'$month','$year') - 24*60*60;print scalar localtime($time);'
}
```

```
function detect_odd() {
echo $1|grep -E '^[1-9]$\ ' >/dev/null 2>&1
}
```

```
function check_dates_in_data() {
newresetday=""
isnewresetday=""
reverselist=""
daysearch=$2
monthsearch=$1
daylist=`nawk '{if($1=="$monthsearch")a[x++]=$2;while(getline)if($1=="$monthsearch"){firstdate=$2;if(firstdate!=nextdate)a[x++]=firstdate;nextdate=$2}}END{for(i=0;i<x;i++)print a[i]}' $f34`
```

```
for i in $daylist
do
reverselist="$i $reverselist"
done
```

```
if [ -z "$3" ] ; then
```

```
for i in $reverselist
do
for((x=1;x<=15;x++)) ; do
if [ $(($10#$daysearch-$x)) -eq $i ] ; then
newresetday=$i
isnewresetday='yes'
break
fi
done
[ "$isnewresetday" = "yes" ] && break
done
```

```
if [ -z "$newresetday" ] ; then
```

```
for i in $daylist
do
for((x=1;x<=15;x++)) ; do
if [ $(($10#$daysearch+$x)) -eq $i ] ; then
newresetday=$i
isnewresetday='yes'
break
fi
done
[ "$isnewresetday" = "yes" ] && break
done
fi
fi
```

```
if [ ! -z "$3" ] ; then
```

```
for i in $daylist
do
for((x=1;x<=15;x++)) ; do
```

```

if [  $((10\#\text{\$daysearch}+\text{\$x})) -eq \text{\$i}$  ] ; then
newresetday= $\text{\$i}$ 
isnewresetday='yes'
break
fi
done
[ " $\text{\$isnewresetday}$ " = "yes" ] && break
done

```

```

if [ -z " $\text{\$newresetday}$ " ] ; then

```

```

for i in  $\text{\$reverselist}$ 
do
for(( $x=1;x\leq 15;x++$ )) ; do
if [  $((10\#\text{\$daysearch}-\text{\$x})) -eq \text{\$i}$  ] ; then
newresetday= $\text{\$i}$ 
isnewresetday='yes'
break
fi
done
[ " $\text{\$isnewresetday}$ " = "yes" ] && break
done

```

```

fi
fi

```

```

if [ -z " $\text{\$newresetday}$ " ] ; then
nextmonth=`find_nextmonth " $\text{\$monthsearch}$ "`
daylist2=`nawk '{if( $\text{\$1}=="\text{\$nextmonth}"$ )a[x++]= $\text{\$2}$ ;while(getline)if( $\text{\$1}=="\text{\$nextmonth}"$ ){firstdate= $\text{\$2}$ ;if(firstdate!=nextdate)a[x++]=firstdate;nextdate= $\text{\$2}$ }}END{for(i=0;i<x;i++)print a[i]}'  $\text{\$f34}$ `
monthin=`date +%b`
lastdate=`find_last_day  $\text{\$monthsearch}$ lnawk '{print  $\text{\$3}$ }`
for i in  $\text{\$daylist2}$ 
do
for(( $x=1;x\leq 10;x++$ )) ; do
if [  $((10\#\text{\$daysearch}+\text{\$x}-\text{\$lastdate})) -eq \text{\$i}$  ] ; then
newresetday= $\text{\$i}$ 
isnewresetday='yes'

```

```

newresetmonth=$nextmonth
isnewresetmonth='yes'
break
fi
done
[ "$isnewresetday" = "yes" ] && break
done
fi

[ -z "$isnewresetday" ] && isnewresetday='no'
[ -z "$isnewresetmonth" ] && isnewresetmonth='no'

}

function resetdays() {

resetdayx=$1
resetdayx=`nawk 'BEGIN{printf "%0.0fn","$resetdayx"}'`

detect_odd $resetdayx
if [ $? -eq 0 ] ; then

$grep "$2 $resetdayx " $f34 >/dev/null 2>&1
if [ $? -ne 0 ] ; then
check_dates_in_data $2 $resetdayx "$3"
resetdayx=$newresetday
else
thisdates="ok"
fi

else

$grep "$2 $resetdayx " $f34 >/dev/null 2>&1
if [ $? -ne 0 ] ; then
check_dates_in_data $2 $resetdayx "$3"
resetdayx=$newresetday
else
thisdates="ok"

```

```

fi

fi
}

function day_ztest() {
day1="$2"
month1="$3"
month2="$4"

if [ $1 -eq 0 ] ; then
[ $5 -eq 1 ] && nawk '/'$month1' '$day1'/{f=1}$1=="$month2"'{print;f=0}f' $f34 > ${f34}_modif
[ $5 -eq 2 ] && nawk '/'$month1' '$day1'/{f=1}$1=="$month2"'{print;f=0}f' $f34 > ${f34}_modif

else
resetdays $day1 $month1

if [ "$thisdates" != "ok" ] ; then

[ "$isnewresetmonth" != "no" ] && month1=$newresetmonth

if [ "$isnewresetday" != "no" ] ; then
day1=$newresetday
else
renewday=`nawk '$1=="$month1"'{print $2;exit}' $f34`
if [ ! -z "$renewday" ] ; then
day1=$renewday
else
echo "No data for the $orgday${fs}${orgmonth} !!"
check_is_nodisk
exit 1
fi
fi
fi
fi

}

```

```

function day2_ztest() {
day2="$2"
month1="$3"
month2="$4"

if [ $1 -eq 0 ] ; then
[ $5 -eq 1 ] && nawk '$1=="$month1"'{f=1}/'$month2' '$day2'/{print;f=0}f' $f34 > ${f34}_modif
[ $5 -eq 2 ] && nawk '$1=="$month1"'{f=1}/'$month2' '$day2'/{print;f=0}f' $f34 > ${f34}_modif

else
resetdays $day2 $month2 day2
day2=$resetdayx

if [ "$thisdates" != "ok" ] ; then

[ "$$isnewresetmonth" != "no" ] && month1=$newresetmonth

if [ "$$isnewresetday" != "no" ] ; then
day2=$newresetday
else
renewday=`nawk '$1=="$month2"'{print $2;exit}' $f34`
if [ ! -z "$renewday" ] ; then
day2=$renewday
else
echo "No data for the $orgfday${fs}${orgfmonth} !!"
check_is_nodisk
exit 1
fi
fi
fi
fi

}

function call_results() {
set_data_file

thisdates="unknownnow"

```

```

case $3 in

1)
orgday="$1";orgmonth="$2"
day1="$1";month1="$2"

if [ "$day1" = "empty" ] ; then
$grep $month1 ${f34} >/dev/null 2>&1
if [ $? -eq 0 ] ; then
nawk '$1=="$month1"' $f34 > ${f34}_modif
else
echo "No data for the ${fs}${month1} !!"
check_is_nodisk
exit 1
fi

else

resetdays $day1 $month1
day1=$resetdayx

if [ "$thisdates" != "ok" ] ; then

[ "$$isnewresetmonth" != "no" ] && month1=$newresetmonth

if [ "$$isnewresetday" != "no" ] ; then
day1=$newresetday
else
$grep $month1 ${f34} >/dev/null 2>&1
if [ $? -eq 0 ] ; then
renewday=`nawk '$1=="$month1"'{print $2;exit}' $f34`
day1=$renewday
else
echo "No data for the $orgday${fs}${orgmonth} !!"
check_is_nodisk
exit 1
fi
fi

```

```

fi

detect_odd $day1
if [ $? -eq 0 ] ; then
nawk '/$month1' '$day1/' $f34 > ${f34}_modif
else
nawk '/$month1' '$day1/' $f34 > ${f34}_modif
fi

fi
;;

2)
orgday="$1";orgmonth="$2"
day1="$1";month1="$2"

if [ "$day1" = "empty" ] ; then
$grep $month1 ${f34} >/dev/null 2>&1
if [ $? -ne 0 ] ; then
echo "No data for the ${month1} !!"
check_is_nodisk
exit 1
fi

else

resetdays $day1 $month1
day1=$resetdayx

if [ "$thisdates" != "ok" ] ; then

[ "$isnewresetmonth" != "no" ] && month1=$newresetmonth

if [ "$isnewresetday" != "no" ] ; then
day1=$newresetday
else
$grep $month1 ${f34} >/dev/null 2>&1
if [ $? -eq 0 ] ; then

```

```

renewday=`nawk '$1=="$month1"{print $2;exit}' $f34`
day1=$renewday
else
echo "No data for the $orgday${fs}${orgmonth} !!"
check_is_nodisk
exit 1
fi
fi
fi

```

```

detect_odd $day1
first_odd=$?

```

```

fi

```

```

;;

```

```

3)

```

```

if [ "$day2" = "empty" ] ; then
grep "$month2" $f34 >/dev/null 2>&1
if [ $? -ne 0 ] ; then
echo "No data for the $orgday${fs}${orgmonth} !!"
check_is_nodisk
exit 1
fi

```

```

if [ "$day1" = "empty" ] ; then
nawk '$1=="$month1"{f=1}$1=="$month2"{print;f=0}f' $f34 > ${f34}_modif

```

```

else

```

```

detect_odd $day1
if [ $? -eq 0 ] ; then
$grep "$month1 $day1" $f34 >/dev/null 2>&1
echo day_ztest X $? X $day1 X $month1 X $month2 X 2
day_ztest $? $day1 $month1 $month2 2
else

```

```
# if day1 is not odd
$grep "$month1 $day1" $f34 >/dev/null 2>&1
day_ztest $? $day1 $month1 $month2 1
echo day_ztest X $? X $day1 X $month1 X $month2 X 2
fi
```

```
fi
```

```
else
```

```
# if day2 is not empty
```

```
if [ "$day1" = "empty" ] ; then
```

```
detect_odd $day2
```

```
if [ $? -eq 0 ] ; then
```

```
$grep "$month2 $day2" $f34 >/dev/null 2>&1
```

```
day2_ztest $? $day2 $month1 $month2 2
```

```
else
```

```
# if day1 is not odd
```

```
$grep "$month1 $day1" $f34 >/dev/null 2>&1
```

```
day2_ztest $? $day2 $month1 $month2 1
```

```
fi
```

```
fi
```

```
fi
```

```
if [ "$day2" != "empty" ] ; then
```

```
if [ "$day1" != "empty" ] ; then
```

```
day2=$1 ; month2=$2
```

```
resetdays $day2 $month2 day2
```

```
day2=$resetdayx
```

```
if [ "$thisdates" != "ok" ] ; then
```

```
[ "$isnewresetmonth" != "no" ] && month2=$newresetmonth
```

```
if [ "$isnewresetday" != "no" ] ; then
```

```
day2=$newresetday
```

```

else
$grep $month2 ${f34} >/dev/null 2>&1
if [ $? -eq 0 ] ; then
renewday=`awk '$1=="$month2"'{print $2;exit}' $f34`
day2=$renewday
else
echo "No data in the $orgfday${fs}${orgfmonth} !!"
check_is_nodisk
exit 1
fi
fi
fi

detect_odd $day2
second_odd=$?

if [ $first_odd -eq 0 ] ; then

if [ $second_odd -eq 0 ] ; then
nawk '/$month1' '$day1'/{f=1}/$month2' '$day2'/{f=0;exit}f' $f34 > ${f34}_modif
else
nawk '/$month1' '$day1'/{f=1}/$month2' '$day2'/{f=0;exit}f' $f34 > ${f34}_modif
fi

else

if [ $second_odd -eq 0 ] ; then
nawk '/$month1' '$day1'/{f=1}/$month2' '$day2'/{f=0;exit}f' $f34 > ${f34}_modif
else
nawk '/$month1' '$day1'/{f=1}/$month2' '$day2'/{f=0;exit}f' $f34 > ${f34}_modif
fi

fi
fi
fi
;;

esac

```

```
}
```

```
function date_convert() {  
nawk -v mon="$1" 'BEGIN{split("Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec",month,"  
");for(i=1;i<=12;i++)if(month[i]==mon)print i}'  
}
```

```
# i guess this is not required for correctly rotates messages files
```

```
function messages_sort() {  
> ${f34}.dates.tmp  
for i in `ls -l /var/adm/messages*`  
do  
date=`nawk '{x=$1 FS $2}END{print x}' $i`  
daydate=`echo "$date"|nawk '{print $2}`  
monthdate=`echo "$date"|nawk '{print $1}`  
numdate=`date_convert $monthdate`  
index=`echo "$i"|sed 's//.*(messages.*)/1/'`=$numdate $daydate"  
echo "$index" >> ${f34}.dates.tmp  
done
```

```
nawk -F=' ' '{a[$NF]=$NF;next}  
function isEmpty(arr,idx){for(idx in arr)return 0;return 1}  
function maxf(a,max){max=(max>a)?max:a;return max}  
function sorted(array,arraymax){for(i in array){  
arraymax2=arraymax;  
split(i,d," ");  
arraymax=maxf(d[1],arraymax);
```

```
if(arraymax>arraymax2){  
j=i;maxday=d[2]  
}
```

```
if(arraymax==arraymax2){  
if(arraymax==d[1]){  
split(j,o," ");  
if(o[2]<d[2]){maxday=d[2];j=i;  
}  
}
```

```

}
}
print a[j];delete a[j]
}
END{while(!isEmpty(a))
sorted(a);
}' ${f34}.dates.tmp > ${f34}.recursive.sorted.messages

}

```

```

function set_data_file() {
detectoldconf $f34 $cc "infunc" "NEW MESSG files"
if [ "$newconff" = "yes" ] ; then
count_mes=`ls -ltr /var/adm/messages* 2>/dev/nullawk 'END{print NR}'`
if [ $count_mes -gt 0 ] ; then
>$f34
# already sorted with ls -l
for i in `ls -l /var/adm/messages*lnawk '{a[++x]=$1;next}END{for(i=x;i>0;i--)print a[i]}' `; do cat $i >>
$f34; done

```

```

##### for later usage maybe ?
#####
# messages_sort
# for i in `nawk -F=' 'NR==FNR{a[$1];next}{if($2 in a)print $1}' ${f34}.recursive.sorted.messages
${f34}.dates.tmp`
# do
# cat $i >> $f34
# done
#####
#####

```

```

else
echo "Cannot be found messages file(s) "
exit 1
fi
fi
}

```

```
dmesg_date_start() {
case $# in
1|2)dmesg_date_parser "$@"
;;
*)dateFormat_error "$@"
;;
esac
}
```

dmesg_data_parser phase

```
function dmesg_parser() {
#nawk 'BEGIN{printf "%#3sn", "$b"}'
if [ ! -z "$1" ] ; then
> "${f34}_modif"
dmesg_date_start "$@"
if [ -s "${f34}_modif" ] ;then
echo "message file(s) are successfully parsed [OK] "
else
echo -en "\033[0;31m"\nNo data to be processed !!\n"
echo -en "\033[0m"
exit 1
fi
else
set_data_file
[ -z "$cc" ] && cc='n'
nawk '1' $f34 > ${f34}_modif
fi
```


#####

dmesg_data_analyzing phase

run works two seq lines for NOW

```
> ${f34}_results
```

```
nawk '
```

```

function parser()
{
for(i=1;i<=NF;i++)
{
if($i~/pci@//$i~/ssd//$i~/sd//$i~/scsi_vhci.*@/)
firstlev=firstlev?firstlev RS FILENAME FS FNR FS $i:FILENAME FS FNR FS $i
if($i~/target/&&$(i+1)~/address:/)
firstlev=firstlev?firstlev RS FILENAME FS FNR FS $i="$i+2):FILENAME FS FNR FS $i="$i+2)
}
}

```

```

/disappeared/{
nextl=1
onelinelev=""
for(i=1;i<=NF;i++)
{
if($i~/PWWN//$i~/_ID/)
onelinelev=onelinelev?onelinelev RS FILENAME FS FNR FS $i:FILENAME FS FNR FS $i
}
print onelinelev RS
}

```

```

/emlxs://qlc://emlx://qla:/{
nextl=1
if(/Loop OFFLINE//Link down//Link OFFLINE/)
{
print FILENAME FS FNR
sub(/.*kern.*]/,"");
#sub(/.*NOTICE/, "");
print FILENAME FS FNR FS $0 RS
}
}

```

```

/scsi://scsi_vhci//genunix:/{
is=0;at=0;op=0;pa=0;of=0;on=0;ide=0;
if(/kern.warning//WARNING//kern.info/)
{
nextl=2

```

```

for(i=1;i<=NF;i++)
{
if($i=="is")is=1
if($i=="at")at=1
if($i=="path")pa=1
if($i=="online")on=1
if($i=="offline")of=1
#if($i~/ide@/)ide=1
}
}

if(is==1)
if(of==1||pa==1)
{
parser()
nextl=1
}

if(at==0&&on==0&&is==0)
parser()

if(nextl==2)
{
getline
if(/error//Error//fail//unable//offlin//timeout//Timeout//NULL//not responding//disappeared/)
{
for(i=1;i<=NF;i++)
{
if($i~/Target//$i~/slot//$i~/Volume/)
secondlev=secondlev?secondlev RS FILENAME FS FNR FS $i FS $(i+1):FILENAME FS FNR FS $i
FS $(i+1)
if($i~/sd//$i~/ssd//$i~/pci@//$i~/lun//$i~/PWWN//$i~/_ID/)
secondlev=secondlev?secondlev RS FILENAME FS FNR FS $i:FILENAME FS FNR FS $i
if($i~/target/)
if($(i+1)~/address:/)
secondlev=RS secondlev RS FILENAME FS FNR FS $i="$$(i+2)
else
secondlev=secondlev?secondlev RS FILENAME FS FNR FS $i:FILENAME FS FNR FS $i

```

```

}
}
else
firstlev=""
}

if(secondlev)
print firstlev RS secondlev RS FILENAME FS FNR RS
else
if(firstlev)
print firstlev RS FILENAME FS FNR RS
firstlev="";secondlev=""
}' ${f34}_modif >> ${f34}_results &
loopx "modified message file(s) are parsing "

}

function hwinfos() {
disk_infos

detectoldconf $f40 $cc "" "NEW DISK/LABEL infos from format"
newconffpreap=$newconff

detectoldconf ${f35}_full $cc "" "new SSD(s) from iostat"
newconffiostat=$newconff

detectoldconf ${f45}_full $cc "" "new SSD(s) from path_to_inst"
newconffpath_to_inst=$newconff

detectoldconf "$f46" $cc "" "NEW device(s) from prtconf"
if [ "$newconff" = "yes" ] ; then
testprtconf
/usr/sbin/prtconf -v >$f46 &
loopx "prtconf is calling "
fi

detectoldconf "$f47" $cc "" "NEW device(s) from prtdiag"
if [ "$newconff" = "yes" ] ; then

```

```
testprtdiag
/usr/sbin/prtdiag -v >$f47 &
loopx "prtdiag is calling "
fi

detectoldconf "$f48" $cc "" "NEW device(s) from scanpci"
if [ "$newconff" = "yes" ] ; then
if [ -x /usr/bin/scanpci ] ; then
/usr/bin/scanpci -v >$f48 &
loopx "scanpci is calling "
else
echo "scanpci cannot be run !! "
screenw 50 "-"
sleep 1
exit 1
fi
fi

detectoldconf "$f49" $cc "" "NEW device(s) from luxadm"
if [ "$newconff" = "yes" ] ; then
if [ -x /usr/sbin/luxadm ] ; then
luxadm probe -p >$f49 &
loopx "luxadm is calling "
else
echo "luxadm cannot be run !! "
screenw 50 "-"
fi
fi

if [ ! -x /usr/sbin/fcinfo ] ; then
echo "fcinfo cannot be run !! "
screenw 50 "-"
exit 1
fi

}
```

```
##### MESSG ANALYZE #####
```

```
function dmesg_analyze() {
echo -en "ec"
>${f34}_temp
nawk 'BEGIN{$80=OFS="=";print RS "Message file(s) analyses " RS $0 RS}' >>${f34}_temp

while read fname nr line
do

if [ ! -z "${fname}${nr}$line" ]; then

case "$line" in
*Target*)Tgt="$line" ;;
*Volume*)Vlm="$line" ;;
*target*|*lun*)tlun=( ${tlun[@]} "$line" ) ;;
*ssd*|*sd*|*disk*)ssdline=( ${ssdline[@]} "$line" ) ;;
*pci@*)pciline="$line" ; issddisk=`echo "$line"|nawk '/ssd/|/sd/|/disk/'` ; [ ! -z "$issddisk" ] &&
ssdline=( ${ssdline[@]} "$line" ) ;;
*slot*)slotline="$line" ;; # request command slot
*Loop*|*Link*)offline="$line" ;; # qllogic or emulex or any hba offline ?
esac

wnr=( ${wnr[@]} $nr )
wfname=$fname

else
#clear
if [ ! -z "$wnr" ] ; then
echo -en '\033[0;31m\n\nError Message part " # RED
nawk 'BEGIN{$80=OFS="-";print RS $0}' $fname #>${f34}_temp
echo -e '\033[1;37m' # DEFAULT
# Error Message Part Print
uniql=`echo ${wnr[@]}|nawk '{for(i=1;i<=NF;i++){if($i!=x)print $i;x=$i}}`
for jj in $uniql
do
nawk 'NR==$jj' $wfname
done #>>${f34}_temp
```

```
# Error Message Part End
screenw 70 "-" #>>${f34}_temp
sleep 2

echo -en "\033[0;31m'
# Error Message Analyse
nawk 'BEGIN{$88=OFS="-";print RS RS $0 RS "ttt ->>>>> ERROR ANALYSIS ->>>>>" RS $0}'
#>>${f34}_temp
echo -en "\033[0m"

if [ ! -z "$pciline" ] ; then
find_pci "$pciline"

if [ ! -z "$tlun" ] ; then
tlun_parser "${tlun[@]}" "$pciline"
fi

if [ ! -z "$Tgt" ] ; then
Targetfinder "$Tgt" "$pciline"
fi

if [ ! -z "$Vlm" ] ; then
Volumefinder "$Vlm"
fi

if [ ! -z "$slotline" ] ; then
slot_finder "$slotline"
fi

fi

if [ ! -z "$ssdline" ] ; then
ssd_pre_parser "${ssdline[@]}"
fi

if [ ! -z "$offline" ] ; then
offline_hba_finder "$offline"
fi
```

```

#if [ $show -eq 1 ] ; then
#less -f -r ${f34}_temp
#else
cat ${f34}_temp
### Results ###
echo -e '\033[1;35m'
nawk 'BEGIN{$70=OFS="-";print RS "ttEND_END_END_END_END" RS $0 RS RS RS }'
sleep 2
#fi

# backup
cat ${f34}_temp >> ${f34}_temp_all
#clear
>${f34}_temp

reset_vals pciline Tgt slotline Vlm offline ctdtolabels
reset_arr_vals ssdline tlun wnr
fi

fi

done < ${f34}_results

[ ! -s "${f34}_temp_all" ] && nawk 'BEGIN{$80=OFS="=";print $0 RS " ==> No Error(s) found .. "
RS $0 }'
>${f34}_temp_all

}

function message_capt() {
message="$1"
nawk -v msg="$message" 'BEGIN{$50=OFS="=";print RS msg RS $0}' >>${f34}_temp
}

function message_end() {
nawk 'BEGIN{$70=OFS="=";print}' >>${f34}_temp
}

```

```

function cfgpath() {
case $# in
1)devcfg=`ls -l /dev/cfg|$grep "$1"|nawk '{print $9}'` ;;
2)devcfg=`ls -l /dev/cfg|$grep "$1"|$grep "$2"|nawk '{print $9}'` ;;
*)echo "Cfg Path Error " ;;
esac
}

```

```

function offline_hba_finder() {
echo "$@"|$grep emlxs >/dev/null 2>&1
if [ $? -eq 0 ] ; then
noemulex=0
hba=`echo "$@"|$sed 's/([^:]*):.*1/' 2>/dev/null`
hbamodel=`echo "$hba"|$sed 's/[0-9]*//g`

hbaportid=`echo "$hba"|$sed 's/[^0-9]*//g`
path_to_inst_chk
if [ "$path_inst" = "ok" ] ; then
pci_hba_path=`$grep "$hbaportid "$hbamodel"" /etc/path_to_inst 2>&1|nawk '{print $1}'`
fi
if [ ! -z "$pci_hba_path" ] ; then
cfgpath "$pci_hba_path"
if [ ! -z "$devcfg" ] ; then
get_fcadm_hba "$devcfg"
check_cfg $devcfg "$hba" "Hba "
else
message_capt "Hba "
echo "$hba cannot be found !!" >>${f34}_temp
message_end
fi
else
message_capt "Hba information part"
echo "Pci path for [ $hba ] cannot be found !! " >>${f34}_temp
message_end
fi
else
noemulex=1
fi

```

```

if [ $noemulex -eq 1 ] ; then
echo "$@" | $grep qlc >/dev/null 2>&1
if [ $? -eq 0 ] ; then
hba=`echo "$1" | $sed 's/. * ([^:]*): .*/1/' 2>/dev/null`
hbamodel=`echo "$hba" | nawk -F '(' '{print $1}'`

echo "$hba" | $grep ', ' >/dev/null 2>&1
if [ $? -eq 0 ] ; then
hbaid=`echo "$hba" | nawk -F '(' '{sub("[0-9]*", "", $2); print $2}'`
hbaportid=`echo "$hba" | nawk -F '(' '{sub("[0-9]*", "", $2); sub("[ ]", "", $2); print $2}'`
else
hbaid=`echo "$hba" | $sed 's/[^0-9]*//g`
fi

path_to_inst_chk
if [ "$path_inst" = "ok" ] ; then
pci_hba_path=`$grep "$hbaportid "$hbamodel"" /etc/path_to_inst 2>&1 | nawk '{print $1}'`
fi
if [ ! -z "$pci_hba_path" ] ; then
if [ -z "$hbaportid" ] ; then
cfgpath "$pci_hba_path"
if [ ! -z "$devcfg" ] ; then
get_fcadm_hba "$devcfg"
check_cfg $devcfg "$hba" "Hba "
else
nawk 'BEGIN{ $80=OFS=" "; print $0 RS " ==> Device Config Path not Found [ '$pci_sd' ]" RS $0 RS }'
>> ${f34}_temp
fi
else
case "$hbaportid" in
0) searchport=`echo "${hbamodel}@0/`
;;
*) searchport=`echo "${hbamodel}@0,${hbaportid}/`
;;
esac
fi
cfgpath "$pci_hba_path" "$searchport"
if [ ! -z "$devcfg" ] ; then

```

```

get_fcadm_hba "$devcfg"
check_cfg "$devcfg" "$hba" "Hba "
else
message_capt "Hba "
echo "$hba cannot be found !!" >>${f34}_temp
message_end
fi
else
message_capt "Hba "
echo "Pci path for [ $hba ] cannot be found !!" >>${f34}_temp
message_end
fi
fi
fi

if [ -z "$hbamodel" ] ; then
message_capt "Hba "
echo "HBA ( not qllogic or emulex ? ) card cannot be found !!" >>${f34}_temp
message_end
fi
}

function check_cfg() {
if [ ! -z "$1" ] ; then
message_capt "Pci/Pciex device port [ $1 ] "
prtconf -v /dev/cfg/$1 >> ${f34}_temp
message_end
else
message_capt "$3"
echo "$2 cannot be found !! " >>${f34}_temp
message_end
fi
}

function get_fcadm_hba() {
hbadev=$1
message_capt "FCadm "
fcadm hba-port|grep -E 'HBA Port|Port ID|OS Device Name|State'|nawk '{a=a RS $0}/HBA

```

```
Port/{a="";hba=$0}/OS Device/{if(/$hbadef/){print hba a;getline;print;exit}}' >>${f34}_temp
message_end
}
```

```
function dev_target_volume() {
echo "$1" | $grep "@v[0-9]*" >/dev/null 2>&1
if [ $? -ne 0 ] ; then
split_pci_path "$2"
cfgpath "$pci_path" "$1"
if [ -z "$devcfg" ] ; then
nawk 'BEGIN{$80=OFS="="};print $0 RS "=="> Device Config Path not Found [ '$pci_sd' ]" RS $0 RS }'
>> ${f34}_temp
else
check_cfg "$devcfg" "$3" "Target "
fi
fi
}
```

```
function Targetfinder() {
#echo Targetfinder "$1"
#iport/sas/scsi/raid controller/disk finder
case "$1" in
"Target 9")dev='@1:'
;;
"Target a"|"Target d")dev='@2:'
;;
"Target b"|"Target e")dev='@4:'
;;
"Target c"|"Target f")dev='@8:'
;;
"Target 323"|"Target 389")dev='@v0:' # assume only one raid volume
;;
*)dev="unknown"; echo "$1 cannot be found !!" >>${f34}_temp
;;
esac
[ "$dev" != "unknown" ] && dev_target_volume "$dev" "$2" "Target "
}
```

```

function scanpci_finder() {
pci_path=$1
prtdiag_pci_device_id=`nawk '/PCI/{a=$0;getline;if($0~"$pci_path"){print a;exit}}' $f47lnawk
'{sub(".*","",$3);print $3}'`
if [ ! -z "prtdiag_pci_device_id" ] ; then
prtdiag_pci_vendor_id=`nawk '/PCI/{a=$0;getline;if($0~"$pci_path"){print a;exit}}' $f47lnawk '{print
$3}'|$sed 's/^[^0-9]([0-9][0-9]*),.*$/1/'`
prtdiag_pci_device_hex=`nawk -v b=$prtdiag_pci_device_id 'BEGIN{printf "%s%#04sn","0x",b}'`
prtdiag_pci_vendor_hex=`nawk -v b=$prtdiag_pci_vendor_id 'BEGIN{printf "%s%#04sn","0x",b}'`

message_capt "Scanpci"
nawk '/vendor '$prtdiag_pci_vendor_hex' device '$prtdiag_pci_device_hex'/{x=$0;;getline a;getline
b;print x RS a RS b;exit}' $f48 >>${f34}_temp
message_end
fi

sas2ircu_pci $pci_path
}

function sas2ircu_pci() {
echo "$1"|$grep -E 'sas@lscsi@' >/dev/null 2>&1
if [ $? -eq 0 ] ; then
sas2ircu_searcher
if [ $? -eq 0 ] ; then
message_capt "Sas2ircu Pci"
$sas2ircuexec listlnawk '/$prtdiag_pci_vendor_id'h/&&/'$prtdiag_pci_device_id'h/{print a RS b RS c
RS d RS $0;exit}/Adapter/{a=$0;getline;b=$0;getline;c=$0}' >${f34}_sas2ircu_searcher
if [ ! -s "${f34}_sas2ircu_searcher" ] ; then
echo "$pci_path not found from sas2ircu !!" >>${f34}_temp
else
cat "${f34}_sas2ircu_searcher" >>${f34}_temp
sas2ircu_find_slots "$1"
fi
message_end
fi
fi
}

```

```

function sas2ircu_find_slots() {
pci_nr=`ls -ltr /dev/cfg|grep "$1"|$grep -v v0|nawk 'END{print NR}'`
[ -z "$pci_nr" ] && echo "Scsi/Sas slots cannot detected !! " || echo "$pci_nr" >${f34}_sas2ircu_slots
}

```

```

function sas2ircu_searcher() {
if [ ! -x "$LOCAL_WORK_DIR/sas2ircu" ] ; then
echo "sas2ircu cannot be find in the [ $LOCAL_WORK_DIR ] directory " >>${f34}_temp
return 1
else
sas2ircuexec=$LOCAL_WORK_DIR/sas2ircu
fi
}

```

```

function sas2ircu_find() {
if [ -s "${f34}_sas2ircu_searcher" ] ; then
>${f34}_sas2ircus_finder
for id in ` $LOCAL_WORK_DIR/sas2ircu list|nawk '/Index/{getline;getline;print $1}'`; do
$sas2ircuexec $id display|nawk '/IR Volume/{print;f=1;next}/Physical device/{f=0}f'
>${f34}_sas2ircus_finder
$grep 'Volume ID' ${f34}_sas2ircus_finder|$grep $2 >/dev/null 2>&1
if [ $? -eq 0 ] ; then
sas_slot_nr=`$grep -i slot ${f34}_sas2ircus_finder|nawk 'END{print NR}'`
pci_nr=`cat ${f34}_sas2ircu_slots`
if [ $sas_slot_nr -eq $pci_nr ] ; then
message_capt "Sas2ircu Volume"
cat ${f34}_sas2ircus_finder >> ${f34}_temp
message_end
message_capt "Sas2ircu Volume Status"
$sas2ircuexec $id STATUS >> ${f34}_temp
message_end
fi
fi
done
fi
}

```

```

function Volumefinder() {

```

```

#case "$1" in
#"Volume 323")voldev='@v0:;cfgpath "$pci_path" "$1" ;;
#"Volume "[0-9]*')voldev="unknown" ;; # '@v0:;cfgpath "$pci_path" "$1"
# i assume its scsi controller because dont know howto find other volumes from console :*(
#*)voldev="unknown" ;;
#esac

```

```

#if [ "$dev" = "unknown" ] ; then
#echo "$1 cannot detected"
## sorry , maybe patch script or trying sas2ircu utilities in the physical HOST ? "

```

```

sas2ircu_find $@
if [ ! -s "${f34}_sas2ircus_finder" ] ; then
message_capt "Sas2ircu Volume"
echo "$1 cannot be found !! " >>${f34}_temp
message_end
fi
}

```

```

function ssd_parser() {

```

```

ssdl="$@"
for ssd in $ssdl
do
case "$ssd" in

*pci@*fp@*ssd@w*|*pci@*scsi@*sd@*|*pci@*disk@*|*pci@*sas@*)
pci_ssd "$ssd"
;;

```

```

fp*ssd@w*)fp=`echo "$ssd"|nawk -F/ '{print $1}'`
fp1=`echo "$fp"|$sed 's/(fp)(.*)/1#2/'`
newssd=`echo "$ssd"|nawk -F/ '{print $2}'`
message_capt "Fiber Port "
nawk -v fp1="$fp1" '/'$newssd'/{x=$0;getline;if(/fpl/)print x}' $f46 >> ${f34}_temp
tport=`echo "$ssd"|nawk -F@ '{print $2}'`
echo "[ Target -> $tport " ] >> ${f34}_temp
message_end

```

```

;;

*ssd@g[0-9]*|*disk@g[0-9]*|*sd@[0-9]*)
case "$ssd" in
*ssd*)dsk="ssd@g" ;;
*disk*)dsk="disk@g" ;;
*sd*)dsk="sd@g" ;;
*)echo "Disk format error !! " ;;
esac
dsk_parser "$ssd" "$dsk"

disk_find "$ctdev"
ctdtolabels "$ctdev"
if [ ! -z "$ctdlabel" ] ; then
message_capt "Disk Label "
echo "Disk label detected [ $ctdev -> $ctdlabel ] " >>${f34}_temp
message_end
else
message_capt "Disk Label "
echo "Disk label not detected [ $ctdev ] !! " >>${f34}_temp
message_end
fi
;;

*disk@w[0-9]*)dsk_parser "$ssd" "disk@w"
if [ ! -z "$ctdev" ] ; then
message_capt "Disk Port "
nawk '/Hardware properties:/{f=1;next}/:/{f=0}f' $f46|nawk
'/product/{getline;p=$0}/vendor/{getline;s=$0}/"$ssd"/{print s FS p RS "Target -> '$ctdev'";exit}'|$sed
"s/.*=('[^']*)' .*=('[^']*)/1 2/" >> ${f34}_temp
message_end
fi
;;

*disk@g[0-9]*|*sd@[0-9]*)preap_formats
ctdev2=`nawk '/"$ssd"/{print x}{x=$2}' $f40 2>/dev/null` ;
if [ ! -z "$ctdev2" ] ; then
write_label "$ctdev2" "Disk "

```

```

fi
;;

*)echo "unknown ssd format !!" >> ${f34}_temp ;
esac

done
}

function write_label() {
ctdtolabels "$1"
[ ! -z "$ctdlabel" ] && echo "Disk detected [ $1 -> $ctdlabel ] " >> ${f34}_temp
message_end
}

function ssd_pre_parser() {
ssdparsedsk=`echo "${@}"|nawk 1 RS=" " |nawk '/ssd@g//ssd@w//disk@//sd@/'`
if [ ! -z "$ssdparsedsk" ] ; then
ssd_parser "$ssdparsedsk"
fi
ssdpar=`echo "${@}"|nawk 1 RS=" " |nawk '/(ssd//(/sd/'`
if [ ! -z "$ssdpar" ] ; then
ssdx=`echo "$ssdpar"|$sed -e 's/[():]*//g`
ssdtoctdfind $ssdx
message_capt "SSD/Disk "
if [ ! -z "$ctddev" ] ; then
echo "Disk detected [ $ssdx -> $ctddev ] " >> ${f34}_temp
message_end
[ -z "$ctdlabel" ] && write_label "$ctddev" "Disk/Label "
else
echo "Disk ctd name not detected [ $ssdx ] !! " >> ${f34}_temp
message_end
fi
fi
}

function lun_finder() {
tlun=`echo "${@}"|nawk 1 RS=" " |$sed -n '/lun/s/lun=(.*)/1/p`

```

```

}

function tport_finder() {
tport=`echo ${@} | nawk 1 RS=" " | sed -n '/target/s/target=(.*)/1/p`
}

function wwnfinder() {
lwwn=`luxadm -e dump_map /devices$1 2>/dev/null | nawk '{a=$4} END {print a}`
}

function isvalidtarget() {
target="$1"
retarget=`echo "$target" | sed 's/,.*//g`
((isvalidtarget++))
if [ $isvalidtarget -eq 1 ]; then
detectoldconf ${f34}_tpwwns $cc "ifunc" "NEW TARGET WWNs for MESSG files"
if [ "$newconff" = "yes" ]; then
for i in `fcinfo hba-port | nawk '/HBA Port/{print $NF}`; do
nawk 'BEGIN{mess="HBA Port";$80=OFS=" ";printf "n%s%sn%sn",mess: ", "$i", $0 }';fcinfo remote-
port -p $i | nawk '/Remote Port/'; done> ${f34}_tpwwns
fi
fi
hbaport=`nawk -v a="$target" '/HBA Port/{hba=$NF;next;next}/Remote/{if(a~$NF){print
hba;exit}f=0}/^ $/|^$/{f=0}f' ${f34}_tpwwns`
if [ ! -z "$hbaport" ]; then
message_capt "Pci HBA WWN Target "
echo "$targetport" >> ${f34}_temp
message_end
fi
}

function find_scsi_initiator() {
devcfg=$1
prtconf -v /dev/cfg/$devcfg 2>/dev/null | nawk '/-port//initiator//iport/{x=$0;getline;a=a?a RS x RS $0:x
RS $0} END {print a}' >> ${f34}_temp
}

function pci_ssd() {

```

```

case $@ in
*fp@*)
pci_fp=`echo "$@"|sed 's/(.*fp@.*)/./1/` # fiber port
cfgpath $pci_fp
if [ ! -z "$devcfg" ] ; then
message_capt "Pci HBA WWN "
prtconf -v /dev/cfg/$devcfg 2>/dev/null |nawk '"name='class'" {getline;if(/fibre/)ok=1 }/initiator-
node/{getline;if(ok==1)print "initiator-port WWN" $0;exit}' >> ${f34}_temp
message_end
prtconf_get "$devcfg"
else
nawk 'BEGIN{$80=OFS="="};print $0 RS "=="> Device Config Path not Found [ '$pci_fp' ]" RS $0 RS }'
>> ${f34}_temp
fi

echo "$@"|grep 'ssd@w' >/dev/null 2>&1
if [ $? -eq 0 ] ; then
targetport=`echo "$@"|sed 's/.*ssd@w(.*)/1/`
invalidtarget "$targetport"
fi

echo "$@"|grep 'disk@w' >/dev/null 2>&1
if [ $? -eq 0 ] ; then
targetport=`echo "$@"|sed 's/.*disk@w(.*)/1/`
invalidtarget "$targetport"
fi

;;

*fibre-channel*)
pci_fibre=`echo "$@"|sed 's/(.*)/sd./1/`
cfgpath $pci_fibre
[ ! -z "$devcfg" ] && prtconf_get $devcfg || nawk 'BEGIN{$80=OFS="="};print $0 RS "=="> Device
Config Path not Found [ '$pci_fibre' ]" RS $0 RS }' >> ${f34}_temp
controller=`sed 's/.*fibre-channel@(.*)/sd./1/`
wcontroller="c$controller"
target=`sed 's/.*fibre-channel@./sd@(.*)/1/`
htarget=`usr/bin/printf "%dn" 0x$target`

```

```

wtarget="t$htarget"
disk=`sed 's/.*/fibre-channel@.*/sd@.*,(.*)/1/`
hdisk=`/usr/bin/printf "%dn" 0x$disk`
wdisk="d$hdisk"
message_capt "Pci SCSI Target "
echo "${wcontroller}${wtarget}${wdisk}" >> ${f34}_temp
message_end
;;

#*ssd@*)
#pci_ssd=`echo $@|sed 's/(.*)/ssd.*1/`
#cfgpath $pci_ssd
#[ ! -z "$devcfg" ] && prtconf_get $devcfg
#;;

*scsi@*|*sd@*|*sas@*)
pci_sd=`echo $@|sed 's/(.*)/sd.*1/`
cfgpath $pci_sd
[ ! -z "$devcfg" ] && find_scsi_initiator $devcfg || nawk 'BEGIN{$80=OFS="="};print $0 RS "==">
Device Config Path not Found [ '$pci_sd' ]" RS $0 RS }' >> ${f34}_temp
;;

*disk@*)
pci_disk=`echo $@|sed 's/(.*)/disk.*1/`
cfgpath $pci_disk
[ ! -z "$devcfg" ] && find_scsi_initiator $devcfg || nawk 'BEGIN{$80=OFS="="};print $0 RS "==">
Device Config Path not Found [ '$pci_disk' ]" RS $0 RS }' >> ${f34}_temp
;;

esac
}

function prtconf_get(){
pci=$1
prtconf -v /dev/cfg/$pci >/dev/null 2>&1
if [ $? -eq 0 ] ; then
nawk 'BEGIN{print "nFull Pci/Pciex device controller [ '$pci' ] " ;$70=OFS="="};print}' >>
${f34}_temp

```

```

prtcnf -v /dev/cfg/$pci >> ${f34}_temp
screenw 70 '-' >> ${f34}_temp
fi
}

function split_pci_path() {
pci_path_parse=`echo "$1"|nawk -F/ '{if($NF~/sd//ssd//disk//fp//iport/){for(i=1;i<NF-1;i++)printf "%s%s",$,i,"/";printf "%sn",$(NF-1)} else print}'`
}

function parse_pci_path() {
pci_dev="$1"
split_pci_path "$pci_dev"
echo "$pci_path_parse"|$grep -E 'sas@lscsi@' >/dev/null 2>&1
# assume the any scsi controller 'v0'
if [ $? -eq 0 ] ; then
scsi_nr=`echo "$pci_path_parse"|$sed 's/.*@(.)$/1/'`
cfgpath "$pci_dev" "$scsi_nr"
else
cfgpath "$pci_dev"
fi
[ -z "$devcfg" ] && nawk 'BEGIN{$80=OFS="="};print $0 RS "=="> Device Config Path not Found [
'$pci_dev' ]' RS $0 RS }' >> ${f34}_temp
}

function find_pci() {
pci_path="$1"
parse_pci_path "$pci_path"

if [ ! -z "$devcfg" ] ; then
prtcnf_get "$devcfg"

##### SCANPCI #####
split_pci_path "$1"

scanpci_finder $pci_path_parse

else

```

```
message_capt "Pci path "  
echo "Pci path [ $pci_path ] cannot detected !! " >> ${f34}_temp  
message_end  
fi  
}
```

```
function gettargetlun() {  
echo BURA OLDU 2 $@  
exit  
case $# in  
1) tport_finder $@ ;;  
2) tport_finder $@ ; lun_finder $@ ;;  
esac  
}
```

```
function target_write() {  
message_capt "Hba/Target port "  
echo "[ WWN -> $1 ] / [ Target -> $2 ]" >> ${f34}_temp  
message_end  
}
```

```
function tlun_parser() {  
check_luxadm  
array=( $@ )  
arrcount=${#array[@]}  
pci_dev=`echo ${array[arrcount-1]}`  
target_lun_port=( ${@/$pci_dev} )
```

```
wwnfinder $pci_dev #lwwn
```

```
echo "${target_lun_port[@]}" | grep lun >/dev/null  
if [ $? -ne 0 ] ; then  
gettargetlun ${target_lun_port[@]}  
if [ ! -z "$lwwn" ] ; then  
message_capt "SAN informaton part "  
target_write $lwwn $tport  
fi  
else
```

```

if [ ! -z "$lwwn" ] ; then
gettargetlun ${target_lun_port[@]}
hexlun=`usr/bin/printf "%dn" 0x$tlun`
tportwwn=`luxadm -e dump_map /devices${pci_dev}lnawk '/$tport/{print $4}`
ldsk=`fcinfo remote-port -s -p $lwwnlnawk '/$tportwwn/{f=1;print;next};Remote Port/{f=0;}f'lnawk
'/LUN: '$hexlun'$/{'while(getline){if($0~"Device
Name"){sub("/.*"/,"",$NF);sub("s[0-9]+","",$NF);print $NF;exit}}}`
message_capt "SAN informaton part "
echo "[ PWWN -> $tportwwn ] / [ WWN -> $lwwn ] " >>${f34}_temp
if [ ! -z "$ldsk" ] ; then
echo "[ Disk -> $ldsk ] / [ Target -> $tport ] / [ Lun -> $tlun ] " >>${f34}_temp
message_end
ctdtolabels $ldsk
if [ ! -z "$ctdlabel" ] ; then
message_capt "Disk "
echo "Disk detected [ $ldsk -> $ctdlabel ] " >>${f34}_temp
else
echo "Disk label not detected [ $ldsk ] !! " >>${f34}_temp
fi
fi
message_end
fi
fi
}

```

```

#function luxadm_fcinfos() {
#luxadm -e portlnawk '!/NOT/{sub(":.*",,"",$1);print $1}' > luxadm_devs
#if [ "$newconffluxadm" = "yes" ] ; then
#while read -r dev
#do
#luxadm -e dump_map $devlnawk
'NR==1 {next;} {a[x++]=$0;pwwn=$4;nwwn=$5}END{ $80=OFS=" ";print "'$dev'" RS
$0;OFS=FS;print "WWN port: "pwwn,"/", "Node WWN port: ",nwwn;for(i=0;i<x-1;i+
#+)print a[i];print "===== END =====}"
#done < luxadm_devs > luxadm_wwns_devs_ports
#fi
#
#nawk '/devices/{devs=$1}/WWN port:/{print dev FS $3}' luxadm_wwns_devs_ports >

```


Block file names are found in /dev/dsk; the names of the raw files are found in /dev/rdisk.

I/O requests (such as lseek(2)) to the SCSI disk must have an offset that is a multiple of 512 bytes (DEV_BSIZE), or the driver returns an EINVAL error.

If the transfer length is not a multiple of 512 bytes, the transfer count is rounded up by the driver.

Partition 0 is normally used for the root file system on a disk, with partition 1 as a paging area (for example, swap).

Partition 2 is used to back up the entire disk.

Partition 2 normally maps the entire disk and may also be used as the mount point for secondary disks in the system.

The rest of the disk is normally partition 6.

For the primary disk, the user file system is located here.

The device has associated error statistics.

These must include counters for hard errors, soft errors and transport errors.

Other data may be implemented as required.

#####

<http://docs.oracle.com/cd/E19253-01/816-5177/scsi-vhci-7d/index.html>

scsi_vhci -> SCSI virtual host controller interconnect driver

The scsi_vhci driver is a SCSI compliant pseudo nexus driver that supports Solaris operating system I/O multipathing services for SCSI-3 devices.

This driver introduces a fundamental restructuring of the Solaris device tree to enable a multipath device to be represented as single device instance rather than as an instance per physical path as in earlier Solaris versions.

The logical units (LUNs) associated multipath SCSI target devices managed by this driver are identified and represented by using the SCSI-3 VPD page (0x83) LUN global unique identifier (GUID) represented as hexadecimal number (64/128 bits)

Symbolic links in /dev/[r]dsk continue to adhere to the cNtNdNsN format.

cN is the logical controller number assigned to this driver instance.

tN is the GUID.

The following is an example of a system with an A5000 storage array:

/dev/rdisk/c4t200000203709C3F5d0s0 -> ../../devices/

scsi_vhci/ssd@g200000203709c3f5:a,raw

...

/dev/rdisk/c4t200000203709C3F5d0s7 -> ../../devices/

scsi_vhci/ssd@g200000203709c3f5:h,ra

...

#

The following is an example of a system with a T300 storage array:

/dev/rdisk/c1t60020F200000033939C2C2B60008D4AEd0s0 ->

../../devices/scsi_vhci/

ssd@g60020f200000033939a2c2b60008d4ae:a,raw

#

/dev/rdisk/c1t60020F200000033939A2C2B60008D4AEd0s7 ->

../../devices/scsi_vhci/

ssd@g60020f200000033939a2c2b60008d4ae:h,raw

The scsi_vhci driver receives naming and transport services from one or more physical HBA (host bus adapter) devices.

To support multi-pathing, a physical HBA driver must have its multipathing enabled and comply with the multipathing services provided by this driver.

#

The scsi_vhci driver supports the standard functions provided by the SCSI interface.

SPARC

SATA --> disk@0,0

SCSI --> sd@0,0

SAS --> disk@w0,0

<http://docs.oracle.com/cd/E19253-01/816-5177/ssd-7d/index.html>

Block file names are found in /dev/dsk; the names of the raw files are found in /dev/rdisk.

#####

```

function preap_formats() {
if [ "$newconffpreap" = "yes" ] ; then
echolformatlnawk '/^ *[0-9][0-9]*./{x=$2 FS $NF;getline;print x RS $0}'!$sed '/>/s/(.*) .*/1
"No_Label"/' > $f40
fi
}

```

```

function dsk_parser() {
ctdev=`echo "$1"!$sed 's/.*'$2'(.)$1/'`
}

```

```

function path_to_inst_chk() {
if [ ! -s "/etc/path_to_inst" ] ; then
echo "/etc/path_to_inst file cannot be found !! "
else
path_inst="ok"
fi
}

```

```

function preapssdtoctd() {
if [ "$newconffiostat" = "yes" ] ; then
iostat -Elawk '/Soft/&&/sd/{print;exit}'!$grep sd >/dev/null 2>&1
if [ $? -eq 0 ] ; then
iostat -Elawk '/Soft/{print $1}' > $f35
iostat -Enlnawk '/Soft/{print $1}' > ${f35}_ctd
nawk '{getline x < ""$f35"" ; print $1 FS x }' ${f35}_ctd > ${f35}_full
fi
fi

```

```

if [ "$newconffpath_to_inst" = "yes" ] ; then
path_to_inst_chk
if [ "$path_inst" = "ok" ] ; then
nawk '/sd/&&/scsi/{sub(/^.*/sd@./, "", $1);sub(/"/, "", $1);gsub(/"/, "", $NF);print $1 FS $NF $(NF-1)}'
/etc/path_to_inst > $f45
nawk 'NR==FNR{a[$1]=$2;next}{for(i in a)if($1~i)print $1,a[i]}' $f45 $f3 > ${f45}_full
fi
fi
}

```

```

function ssdtoctdfind() {
preapssdtoctd
#ssddev=`echo "$1"|$grep -Ev 'ssd@lssd '| $sed 's/.*(ssd[0-9]*).*/1/g`
ctddevs=`nawk -v ssddev=$1 '$2==ssddev{print $1}' ${f35}_full ${f45}_full 2>/dev/null`
ctddev=`echo "$ctddevs"|nawk '!a[$1]++`
#ctdtolabels $ctddev
}

```

```

function ctdtolabels() {
disk_infos
ctdlabel=`nawk -v ctd=$1 '$1~ctd{print $2}' $f3`
}

```

```

function slot_finder() {
echo "$1 is a virtual app/io sas/scsi slots" >> ${f34}_temp
screenw 70 '-' >>${f34}_temp
}

```

```

function sedconvert() {
newl=$(echo "$1"|$sed 's///\//g')
echo "has a bug -> `echo "$1"|$sed 's///\//g`"
}

```

```

function reset_vals() {
for i in $@
do
eval $i=""
done
}

```

```

function reset_arr_vals() {
for i in $@
do
eval $i=()'
done
}

```

```

function disk_find() {

```

```

disk=$1
fulllogicpath=`nawk -F: '/'$disk'/{lpath=$0;getline;x1=$0;getline;print nwwn RS lpath RS x1 RS $0
;exit}{nwwn=$0}' $f49 2>/dev/null`
if [ ! -z "$fulllogicpath" ]; then
message_capt "Full Disk Path Information part"
echo "$fulllogicpath" >>${f34}_temp
nawk 'BEGIN{$50=OFS="=";print $0 }' >>${f34}_temp
logicpath=`nawk -F: '/'$disk'/{print $NF;exit}' $f49`
prtinfos "$logicpath"
else
message_capt "Logical Path "
echo "Disk logical path cannot be found for [ "$1" ] !! " >> ${f34}_temp
message_end
fi
}

function prtinfos() {
check_luxadm
message_capt "Logical Path "
luxadm display $1nawk '/Vendor/{x=$NF}/Product/{x=x_"$NF}/Controller//Device Address//port
WWN/{a=a?a RS $0:$0}/State/{if($2!~/ONLINE//online/)print x " -> '$logicpath' " RS a RS $0 ; else
a="" }' 2>/dev/null >> ${f34}_temp
# try to find paths from prtconf
prtconf -v $1nawk '/Path /{x=$0;getline;print x RS $0}' >> ${f34}_temp
message_end
}

# MESSG FUNCTION STARTS #

function dmesg_analyze_start() {
LOCAL_WORK_DIR="/usr/local/bin"
f3=${LOCAL_TEMP_DIR}/disksinfos_full_labeln
f34=${LOCAL_TEMP_DIR}/dmesgs
f35=${LOCAL_TEMP_DIR}/iostats_ssd
f40=${LOCAL_TEMP_DIR}/preapformats
f45=${LOCAL_TEMP_DIR}/path_inst_ssd
f44=${LOCAL_TEMP_DIR}/dmesg_formats
f46=${LOCAL_TEMP_DIR}/prtconf.tmp

```

```
f47=${LOCAL_TEMP_DIR}/prtdiag.tmp
f48=${LOCAL_TEMP_DIR}/scanpci.tmp
f49=${LOCAL_TEMP_DIR}/luxadm_probe.tmp
dmesg_parser "$@" # $@ --> specific dates
hwinfos
dmesg_analyze
}
```

```
function parametersuniq() {
parameters="$(echo "$@"|nawk ' $1=$1' RS= OFS="n"|nawk '!a[$1]++)'"
#echo $parameters|grep "noecho" >/dev/null
#[ $? -eq 0 ] && noecho=1 || noecho=0
}
```

```
function paramscontrol() {
for i in $@
do
case "$i" in
*#[#,%(:);,=<=>[\`{ }~]=]*) echo -e "unrecognized character(s) detected -> [ $i ] n" ; exit 1 ;;
# has a bug ? *#[#%( )+,-/;,<=>[\`{ }~]=]*) echo -e "unrecognized character(s) detected -> [ $i ] n" ; exit 1
;;
esac
done
}
```

```
function executecmd() {
case $1 in
1) cal_hba_ports;hba_states ;;
2) hba_info ;;
3) hba_pci;hba_lun_path "$parameters" ;;
4) disk_mpaths "$parameters" ;;
5) disk_format "$parameters" ;;
6) detect_wwn_disk "$parameters" ;;
7) detect_pwwn_disk "$parameters" ;;
8) list_luns_disklbl "$parameters" ;;
9) non_mpaths ;;
10) disk_errors "$parameters" ;;
11) disk_hba "$parameters" ;;
```

```

12) inq_sym_wwn "$parameters" ;;
13) inq_vendor "$parameters" ;;
14) inq_hba "$parameters" ;;
15) dmesg_analyze_start "$dates" ;;
# [qQxX]*) resetshll ; exit 1 ;;
*) trap 'resetshll' 1 2 9 15 ;;
esac
resetshll
}

```

```

function preexecutecmd() {
s=$1;shift
params="$@"
paramscontrol "$params"
if [ ! -z "$params" ] ; then
parametersuniq "$params"
fi
executecmd $s
}

```

```

function dmesg_only() {
show=0
if [ $# -eq 1 ] ; then
executecmd $1
else
s=$1
shift
dates="$@"
executecmd $s
fi
}

```

```

function menu() {
((define_varp++))
[ $define_varp -eq 1 ] && define_var
if [ "$1" = "show" ] ; then
show=1
read -p "Select Storage Informations [1-15] ... or [q/x] ? " s parameters;

```

```

case $s in
[0-9]|1[0-4]) selectmenus $s
preexecutecmd $s "$parameters"
echo
;;
15) dates="$parameters"; selectmenus $s ; executecmd $s
;;
[qQxX]*) resetshll ; exit 1
;;
*) echo -e "\33[0;31m' "n==> Please use only [1-15] menu(s) and Parameters ( OPTIONAL ) !! n"
resetshll
menu show
;;
esac

else
show=0

case $1 in
[0-9]|1[0-4]) # selectmenus $1
s=$1;shift
preexecutecmd $s "$@"
;;
15) dmesg_only "$@"
# selectmenus $1
resetshll
;;
esac
fi
}

function selectmenus() {
echo
echo -e "\33[1m -> Selected Item [ $1 ]"
echo
echo
}

```

```

function menu_0() {
[ -z $1 ] && go show
while :
do
resetshll;
menu_view
menu show
keycontinue;

#x=1;
#for((i=0;i<=1;i++))
#do
#printf "zzz.r"
#menu_clearer 1 $x
#((x++))
#done

#waitm

cc="n" # " default config is 'no' thats use old files "
done
}

function error_msg () {
resetshll
echo -e "Correct usage -> $0 [ 'system rescanning option -> y(y)/n(y)' ] [0-15] [Optional -> Parameter(s)]
n" ; exit 1 ;
}

function menu_view() {
echo -en "ec"
nawk 'BEGIN{ $80=OFS="="; print $0 RS }'
echo
echo " ----- "
echo " | SOLARIS SPARC MPATH/HBA/SAN/DISK INFORMATIONs | "
echo " ----- "
echo " 1-) HBA states [online/offline]: "
echo " 2-) HBA-LUN-PORT General Informations: "

```

```

echo " 3-) HBA-LUN-PORT-PATH Specific Informations: (WWN) "
echo " "

echo " 4-) Storage[EMC/IBM/HITACHI] Disks [MPxIO]: (LABEL_NAME or DISK ID) "
echo " 5-) Storage[EMC/IBM/HITACHI] Disks [LABEL -> DISK ID MATCH]: (LABEL_NAME) "

echo " 6-) Storage[EMC/IBM/HITACHI -> Host WWN ]: (WWN) "
echo " 7-) Storage[EMC/IBM/HITACHI -> Remote WWN ]: (PWWN) "
echo " 8-) Storage[EMC/IBM/HITACHI -> WWN_PWWN_LUN ]: (LABEL_NAME or DISK ID) "

echo " 9-) Non MPxIO Disks List: "
echo " 10-) Disks Errors: (LABEL_NAME) "
echo " 11-) Extendend Infos [LUN-WWN-RWWN-NWWN-PATHS (ONLINE/OFFLINE)]:
(LABEL_NAME or DISK ID) "
echo " "

echo " 12-) Inq Infos [[EMC/IBM/HITACHI Device-Serial-WWN-Label]: (LABEL_NAME or DISK
ID) "
echo " 13-) Inq Infos [DEVICE-VEND-PROD-REV-SER NUM-CAP(kb)]: (DISK ID) "
echo " 14-) Inq HBA Infos [HBA-Vendor-Port Type-Speed]: (Host WWN) "
echo " "

echo " 15-) MESSAGES ERROR ANALYZER: '[dd][fs][mm][,;][dd][fs][mm]' "
echo " "
echo " [ ALL parameters is OPTIONAL (AVALIABLE PARAMETERS SPECIFIED in the
PARENTHESES) ]"
nawk 'BEGIN{$80=OFS="=";print $0 RS }'
echo "Usage -> $0 [y|ny|yy|n : Re-Scan the system force [y]es or [n]o [menu_nr: (1-15)] [parameters]"
nawk 'BEGIN{$80=OFS="=";print $0 RS }'
}

function cccheck() {
[ -z "$1" ] && echo "New/Old Files( recent results ) MUST BE Selected" && exit 1
[ "$1" = "x" ] && exit 0
[ "$1" = "X" ] && exit 0
yncheck "$1"
if [ $? -ne 0 ] ; then
nawk 'BEGIN{$80=OFS="=";print $0 RS " ==> invalid input !! ['$1'] " RS $0 RS }'

```

```
exit 1
fi
}
```

```
function go() {
resetshll
if [ "$1" = "show" ]; then
read -p "System Re-Scanning -> [y(y)/n] ? " cc
cccheck "$cc"
else
cc=$1
fi
}
```

```
function detect_system() {
fp="$LOCAL_TEMP_DIR/platf"
if [ ! -s "$fp" ]; then
nawk 'BEGIN{$80=OFS="=";print $0 RS "System Platform "}' > $fp
platf=`prtconf -bnawk '/banner/{print $NF;exit}' 2>/dev/null `
[ -z "$platf" ] && prtdiag lnawk '/System/{print "Platform : " $NF;exit}' >> $fp || echo "Platform :
$platf" >> $fp
nawk 'BEGIN{$80=OFS="=";print $0 RS }' >> $fp
fi
#echo -e "Platform -> $(<$fp)n"
cat $fp
sleep 1
clear
}
```

```
function test_hba() {
for hbawwn in `fcinfo hba-portlnawk '/HBA Port/{print $NF}`
do
driver=`fcinfo hba-port $hbawwnlnawk '/Driver Name/{print $NF}`
case "$driver" in
emlx)echo "Emulex(Non-Oracle) card detected .. [$driver] " ;;
emlxs)echo "Emulex(Oracle) card detected .. [$driver] "
blinked " Some results may be wrong !!" ;;
*qla*|*qlc*)echo "Qlogic cards detected .. [$driver] " ;;
```

```

*)echo "Some FC cards are not detected .. [$driver] ? " ;;
esac
done
resetshll
sleep 1
clear
}

```

```

function detect_last_work_date() {
last=`cat $LOCAL_TEMP_DIR/last_work_date`
now=`date +%s`
timegap=`nawk -v now=$now -v last=$last 'BEGIN{print now-last}`
# i assume , detect the system for '6 hours' periods
sixhours=21600
if [ $timegap -gt $sixhours ] ; then
detect_system
test_hba
rm -f "$LOCAL_TEMP_DIR/last_work_date"
fi
}

```

```

function lets_go() {
orgparams="$@"
LOCAL_TEMP_DIR="/tmp/solaris_system_check"
if [ ! -d "$LOCAL_TEMP_DIR" ] ; then
mkdir -p $LOCAL_TEMP_DIR
if [ $? -ne 0 ] ; then
nawk 'BEGIN{${0}=OFS="=";print $0 RS " ==> tmp directory cannot created !! " RS $0 RS }'
exit 1
else
screenw 80 '-'
echo "Script is using temporary space on the ' $LOCAL_TEMP_DIR ' directory "
echo "if you wish later you can remove the files from dir for FRESH results "
screenw 80 '-'
fi
fi
if [ ! -s "$LOCAL_TEMP_DIR/last_work_date" ] ; then
echo `date +%s` > $LOCAL_TEMP_DIR/last_work_date

```

```
detect_last_work_date
else
detect_last_work_date
fi
```

```
if [ $# -eq 0 ] ; then
menu_0
fi
```

```
if [ $# -eq 1 ] ; then
case $1 in
ylnylyln) go $1 ; menu_view ; menu show ; [ "$keycont" != "no" ] && keycontinue ; menu_0 1 ;;
[0-9]|1[0-4]) menu $@ ;;
15) menu $1 ;;
*) error_msg ;;
esac
fi
```

```
if [ $# -gt 1 ] ; then
no=0
```

```
case $1 in
ylnylyln) no=1 ; go $1 ;;
[0-9]|1[0-4]) if [ $no -eq 1 ] ; then shift ; menu $@ ; else menu $@ ; fi ;;
15) if [ $no -eq 1 ] ; then shift ; dmesg_only $@ ; else dmesg_only $@ ; fi ;;
*) error_msg ;;
esac
```

```
case $2 in
[0-9]|1[0-4]) if [ $no -eq 1 ] ; then shift ; fi ; menu $@ ;;
15) if [ $no -eq 1 ] ; then shift ; fi ; dmesg_only $@ ;;
*) if [ $no -eq 1 ] ; then error_msg ; fi ;;
esac
```

```
fi
}
```

```
trap " 2 3 15 ;
```

lets start to control the NEEDED tools
check_tools

lets go :)
lets_go "\$@"

Posted - Tue, Jan 11, 2022 8:12 AM. This article has been viewed 7716 times.

Online URL: <http://kb.ictbanking.net/article.php?id=727>