```
*************************************************************
        Shell Script (KornShell)
*************************************************************

#!/bin/ksh


# --- Including directories in the PATH ---
export PATH=/home/bringas/Tools:$PATH
. /home/bringas/kshrc.sh



# --- Definition of variables ---
a=-60
b=15
file_in=/WSmounts/d2/goni/ALTIMETRY/MAPS_AVISO/DTREF/19930106.gz
(( a = 1 ))                     # this is valid for numerical variables
(( a = a + 1 ))                 # numerical operations
(( a = b + c ))                 # numerical operations, also * and /

$var                            # this is the value of the variable var

#$str is the same that ${str}
echo $str ${str}                # print the same since two times
echo ${str#???}                 # print the string without the 3 first characters
echo ${str%???}                 # print the string minus without the last 3 characters
length=${#str}                  # length is the length (size) of the string str



# --- Initializing a file ---
cat /dev/null > ./filename



# --- The IF statement ---
str="19991131"
echo $str ${str%??}
if (( ${str%??} > 199711 )) && (( ${str%??} < 200206 )) || ((${str%??}==200001))   # use (( )) for
numbers, spaces are not important
then
  echo "both conditions are true"
else
  echo "at least one condition is false"
fi

if [[ ${str%??} = 199711 ]]                     # use [[ ]] for strings, spaces are important!
then
  echo "the condition is true"
else
  echo "the condition is false"
fi
```

```
if [[ -n $str ]] then
   echo "will enter here (TRUE) if the length of the string $str is > 0"
else
   echo "will enter here (FALSE) if the length of the string $str is 0 (if there is no value
associated to $str)"
fi

if [[ -f $file ]] then
   echo "enter here (TRUE) if $file is a regular file (even if it is empty)"
else
   echo "enter here (FALSE) if $file is not a regular file (if $file does not exist)"
fi

if (( a < 0 ))
then
   echo "a < 0"
elif (( a == 0 ))
then
   echo "a = 0"
else
   echo "a > 0"
fi


# --- The for loop
for yr in 2000 2001 2002 2003 2004 2005 2006 2007
do
   mthlist=`cat filemth_$yr`
   for mth in $mthlist
   do
      zcat sst_data.gz | awk ' ( $1==yr && $2==mth ) { print $3 } ' yr=$yr mth=$mth | sort -n | uniq
> "filedy_"$mth"_"$yr
   done
done



# --- The while loop ---
while read rec; do
   DATE=`basename $rec .txt.gz`
   zcat $rec > filename
   if (( $? == $zero )) then
      (( Xmax = Xmx ))
      if (( Xmin < Xmax )) then
         awk ' ( $1 >= Xmi - 5 && $1 <= Xma + 5 && $2 >= Ymi - 5 && $2 <= Yma + 5 && $3 > Tm ) {
print $0 } ' Xmi=$Xmin Xma=$Xmax Ymi=$Ymin Yma=$Ymax Tm=$Tmin $DIR/tmpdel1 >> $DIR/tmpdel2
      else
         awk ' ( ( $1 >= Xmi - 5 && $1 <= 180 && $2 >= Ymi - 5 && $2 <= Yma + 5 ) || ( $1 >= -180 &&
$1 <= Xmax + 5 && $2 >= Ymi - 5 && $2 <= Yma + 5) && $3 > Tm ) { if ( $1 > 0 ) print $0;
else print $1 + 360, $2, $3 } ' Xmi=$Xmin Xma=$Xmax Ymi=$Ymin Yma=$Ymax Tm=$Tmin
$DIR/tmpdel1 >> $DIR/tmpdel2   # we read the data and them transform it from -180/180 to
0/360
         (( Xmax = Xmax + 360 ))
      fi
   fi
done < ./list
```

```
i=1
imax=10
while (( i <= imax ))        # using while loop instead of for loop
do
   x=`echo $xres | awk ' {print $1*2*n} ' n=$i`
   (( i = i + 1 ))
done



# --- The case statement ---
$mth                              # this contain the month information with one or two digits
typeset -i mth_length=${#mth}     # after this dimth is the month with 2 digits always
if (( mth_length == 1 )) then
   dimth="0$mth"
else
   dimth=$mth
fi
case $dimth in                   # after this, dmth is the month in Mmm format
   "01") dmth="Jan";;
   "02") dmth="Feb";;
   "03") dmth="Mar";;
   "04") dmth="Apr";;
   "05") dmth="May";;
   "06") dmth="Jun";;
   "07") dmth="Jul";;
   "08") dmth="Aug";;
   "09") dmth="Sep";;
   "10") dmth="Oct";;
   "11") dmth="Nov";;
   "12") dmth="Dec";;
esac



# --- Find ---
find /WSmounts/d2/goni/ALTIMETRY/MAPS_AVISO/DTREF/ -name "????????.gz" > ./list_SHA
find /WSmounts/d1/SST/ -name "????????.txt.gz" > ./list_SST



# --- Extracting data ---
zcat $file_in | awk '{print $2, $1, $3}' > filename
```

```
# --- Arrays in Awk ---
awk ' BEGIN { FS = "," } ( NR > 8 && substr($8,2,2) < 50 && $10 < 40 && $10 > 25 ) { if ($5=="N")
slat=1; else slat=-1; if ($7==W) slon=-1; else slon=-1; if (substr($8,1,1)=="+") stemp=1; else
stemp=-1; c+=1; print dt, $2, $6*slon, $4*slat, substr($8,2,6)*stemp, $10*1, c } ' dt=$dt file_in >
file_out

awk ' {c+=1; printf"%.6f\t%.6f\t%.0f\n", $1, $2, c} ' file_in | awk ' { X[$3]=$1; Y[$3]=$2; tot+=1 }
END {
    for (i=2; i<=tot-1; i++)
        {printf"%.6f\t%.6f\n", X[i]-(X[i]-X[i-1])/2, Y[i];
        printf"%.6f\t%.6f\n", X[i]+(X[i+1]-X[i])/2, Y[i]};
        printf"%.6f\t%.6f\n", X[tot]-(X[tot]-X[tot-1])/2, Y[tot];
        printf"%.6f\t%.6f\n", X[tot]+(X[tot]-X[tot-1])/2, Y[tot];
        printf"%.6f\t%.6f\n", X[tot]+(X[tot]-X[tot-1])/2, 0;
        printf"%.6f\t%.6f\n", X[1]-(X[2]-X[1])/2, 0
    } ' > file_out

cat ./file_in | awk ' { X[$3]=$1; Y[$3]=$2; n+=1 } END { NC=0;
        for (i=1; i<=n; i++)
        {
            if ( X[i]==999 )
            {
                NC=NC+1;
                NCndx[NC]=i
            };
        };
        Max=0;
        Dstndx=0;
        for (i=1; i<NC; i++)
        {
            Dst[i]=NCndx[i+1]-NCndx[i];
            if ( Dst[i]>Max )
            {
                Max=Dst[i];
                Dstndx=i;
            };
        };
        for (i=NCndx[Dstndx]+1; i<NCndx[Dstndx+1]; i++)
        {
            print X[i], Y[i]
        };
    } ' > ./file_out


# --- Creating a file inside the script ---
cat >> filename << EOF    # you can include empty lines but not comments
first line of data
second line of data
more data
The end of the data

1
2   3
4   5   6
EOF
```

```
# --- Converting images ---
convert -rotate 90 -trim +repage -quality 100 ./file.ps ./file.jpg
convert ./file.jpg ./file.gif

gifmerge -l30 ./file_*.gif > ./animation.gif

montage -tile 3x1 -geometry 250x200+3 ./file1.jpg ./file2.jpg ./file3.jpg ./file.gif


# --- Using join and nunion
# file1 and file2 have 3 columns each, we want to joint for the first two columns and then operate in
the third (sum column 3 of both files):
join ./file1 ./file2 | awk ' ( $2 == $4 ) { print $1, $2, $3 + $5 } ' > ./file3
nunion -n2 ./file1 ./file2 | awk ' ( NF == 4 && $4 != 0 ) { print $1, $2, $3 + $4 } ' > ./file3


# --- From Julian to Gregorian date -----
# if jdt is the Julian day, to convert it into Gregorian date (gdt) in the format yyyymmdd :
gdt=`echo $jdt | julgre -j2448622 | awk '{ yr = $1; if ( length($2) == 2 ) mth = $2; else mth =
"0"$2; if ( length($3) == 2 ) dy = $3; else dy = "0"$3; printf"%4s%2s%2s", yr, mth, dy }'`


#--- Using functions -----
#!/bin/ksh

function absolute_value
{
   a=$1
   b=$2
   abs_val=`echo $a $b | awk '{ print sqrt($1^2 + $2^2) }'`
   return
}

function absolute_value2
{
   a=$1
   b=$2
   echo $a $b | awk '{ print sqrt($1^2 + $2^2) }'
   return
}

Vx=10
Vy=20
Wx=30
Wy=40

absolute_value $Vx $Vy
(( abs_V = abs_val ))

absolute_value $Wx $Wy
abs_W=$abs_val
abs_V=`absolute_value2 $Vx $Vy`
return
```

```
# --- ending...
rm -f ./tmpfile
return




**************************************************************
        GMT
**************************************************************

# for help go to      http://gmt.soest.hawaii.edu/gmt/gmt_man.html

# --- to plot a function y=f(x). The file file2d has two columns: x, y:
psxy ./file2d -X1 -Y1 -JX7 -R$xmin/$xmax/$ymin/$ymax -B:."Title":Sn5a10g10/We5a10g10 -Sc0.1 -
W1.5p,255/255/255 -G255/0/0 -K > ./file.ps     # if this is your first GMT command for this plot, or
psxy ./file2d -X0 -Y0 -J -R -Sc0.1 -W1.5p,255/255/255 -G255/0/0 -O -K >> ./file.ps     # if this is
not the first command -W outline of the symbols, -G fill symbols

# --- to make a map of the function z=f(x,y). The file file3d has 3 columns: x, y, z:
makecpt -Cno_green -T$minz/$maxz/$zstep -Z -D > ./file.cpt     # -Z continuous, -D extremes

xyz2grd ./file3d -G./file3d.grd -R$xmin/$xmax/$ymin/$ymax -I$xres/$yres

grdimage ./file3d.grd -C./file.cpt -X$Xpos -Y$Ypos -JQ0/$Psz -R$xmin/$xmax/$ymin/$ymax -
B:."Title":Sn5a10g10/We5a10g10 -K > ./file.ps    #grdimage will plot the image

grdcontour ./file3d.grd -X0 -Y0 -J -R -C10 -A10 -W1p,50/50/50 -Q100 -O -K >> ./file.ps   #grdcontour
to plot contour -W for the contours

echo $lon $lat | psxy ./file2d -J -R -Sc0.1 -W1.5p,255/255/255 -G255/0/0 -O -K >> ./file.ps

pscoast -J -R -G200/200/200 -C255/255/255 -W -O -K >> ./sst_ctr_$DATE.ps       # -G for dry areas, -C
for lakes, -W draw coastlines

psscale -X3 -Y-1 -D0.5/0.5/7/0.2h -Bn0.5a1g1:"|grad(SST)| [@+0@+C/deg]":/:"   ": -C./file.cpt -O >>
file.ps   # -Dxpos/ypos/length/width[h]. In this example, this is your last line in this example, you
don't put -K

# --- Other GMT programs:
grdgradient
grd2xyz
pslegend
minmax
...
```