Wrapper script for ImageJ

Description

A command line wrapper script to launch ImageJ on Linux / Mac **os** / Cygwin platforms, with the following features

- Open images from the command line
- Launch macros / execute commands
- Send images / commands to running ImageJ windows
- Set available memory

Author

Jonathan Jackson (jjackson a.t familyjackson dot net)

Change List

- 2008-12-04: First posted on the ImageJ Wiki
- 2009-08-21: Now works with MacOS X (tested on 10.5). Fixed bug with assignment of lock files to open ports.

Limitations

- Can't open files in existing ImageJ panel on MacOS X
- Not tested on Cygwin

Usage

```
#!/bin/bash
# A wrapper script to launch imagej from the UNIX command line
# Images given as arguments will be opened, macros may also be given as
arguments
# Looks for macros in the imagej macro directory
#
# This program is free software, but comes with no warrenty or guarantee
# send bug reports or feedback to jjackson at familyjackson dot net
# Author: Jon Jackson
# Last modified date: $Date: 2008-12-04 18:38:47 +0000 (Thu, 04 Dec 2008) $
# $Revision: 64 $
#
```

```
# INSTALLATION INSTRUCTIONS
#
### WARNING ############################################################
# This file won&#039;t run if edited in &#039;Notepad&#039; or similar
programs
# that don&#039;t support unix new line characters
########################################################################

# Source location: http://rsb.info.nih.gov/ij/download/linux/unix-script.txt
# 1) Save this script in the ImageJ directory as &#039;imagej&#039;
# 2) Modify path variables according to your system
# 3) Give the new file execute permission
# 4) Be sure the &#039;imagej&#039; wrapper is in the &#039;PATH&#039;

### DEPENDENCIES #######################################################
# readlink: This script works better with GNU readlink, which is not
#           installed by default on MacOS (or Solaris 8)
# MacOS X:  readlink can be installed with &#039;MacPorts&#039;:
#             Install MacPorts and Apple xCode, then run
#             &#039;sudo port install coreutils&#039; and set the PATH
variable appropriately

# ALL:      If no version of readlink is avaliable, hard code the ij_path and
ij_jar_paths
#           sending images to a running ImageJ panel may not work (around
line 357)
########################################################################

# setup environment
set +u # don&#039;t give error for unset variables (matters for environment
variables)
shopt -s extglob # allow extended pattern matching
shopt -s expand_aliases

# Detect readlink version
# if GNU readlink is known to be installed, this code can be replaced by
&quot;alias ReadLink=&#039;readlink -f&#039;&quot;
alias ReadLink=&#039;readlink&#039; # Default
if [[ $(uname) == Darwin ]] &amp;&amp; which greadlink &amp;&gt;/dev/null ;
then
        # Using GNU readlink on MacOS X
        alias ReadLink=&#039;greadlink -f&#039;
elif [[ -f $(which readlink) ]] &amp;&amp; readlink --version | grep
coreutils &amp;&gt;/dev/null ; then
        alias ReadLink=&#039;readlink -f&#039;  # use GNU readlink
else
        alias ReadLink=&#039;echo&#039;
        echo &quot;Please install GNU readlink. Symbolic links may not be
resolved properly&quot; &gt;&amp;2
fi
```

```
############ SITE SPECIFIC VARIABLES ##########################
# Trailing / is not required for path variables
# IMAGEJ PATH - production installation
#ij_path=&#039;/local/ImageJ&#039;
ij_path=&quot;$(dirname $(ReadLink
```

```
#!/bin/bash # A wrapper script to launch imagej from the UNIX command line # Images given as arguments will be opened, macros may also be given as arguments # Looks for macros in the imagej macro directory # # This program is free software, but comes with no warrenty or guarantee # send bug reports or feedback to jjackson at familyjackson dot net # Author: Jon Jackson # Last modified date: $Date: 2008-12-04 18:38:47 +0000 (Thu, 04 Dec 2008) $ # $Revision: 64 $ # # INSTALLATION INSTRUCTIONS # ### WARNING ############################################################ # This file won't run if edited in 'Notepad' or similar programs # that don't support unix new line characters ######################################################################## # Source location: http://rsb.info.nih.gov/ij/download/linux/unix-script.txt # 1) Save this script in the ImageJ directory as 'imagej' # 2) Modify path variables according to your system # 3) Give the new file execute permission # 4) Be sure the 'imagej' wrapper is in the 'PATH' ### DEPENDENCIES ##################################################### # readlink: This script works better with GNU readlink, which is not # installed by default on MacOS (or Solaris 8) # MacOS X: readlink can be installed with 'MacPorts': # Install MacPorts and Apple xCode, then run # 'sudo port install coreutils' and set the PATH variable appropriately # ALL: If no version of readlink is avaliable, hard code the ij_path and ij_jar_paths # sending images to a running ImageJ panel may not work (around line 357) ######################################################################## # setup environment set +u # don't give error for unset variables (matters for environment variables) shopt -s extglob # allow extended pattern matching shopt -s expand_aliases # Detect readlink version # if GNU readlink is known to be installed, this code can be replaced by "alias ReadLink='readlink -f'" alias ReadLink='readlink' # Default if [[ $(uname) == Darwin ]] && which greadlink &>/dev/null ; then # Using GNU readlink on MacOS X alias ReadLink='greadlink -f' elif [[ -f $(which readlink) ]] && readlink --version | grep coreutils &>/dev/null ; then alias ReadLink='readlink -f' # use GNU readlink else alias ReadLink='echo' echo "Please install GNU readlink. Symbolic links may not be resolved properly" >&2 fi ############ SITE SPECIFIC VARIABLES ########################## # Trailing / is not required for path variables # IMAGEJ PATH - production installation #ij_path='/local/ImageJ' ij_path="$(dirname $(ReadLink $0))" # IMAGEJ PATH - development installation (optional) ij_path_dev='/Users/jjackson/ImageJ' # JAVA PATH # assumes executable is ${java_home}/bin/java # set java_home variables ='' to use JAVA_HOME environment variable # Get a sensible default if which java > /dev/null ; then java_home="$(dirname $(ReadLink $(which java)))" fi # The following allow platform specific defaults for the java path # these take preference over the default if set if [[ -d /usr/java/jdk1.6_x64 ]] ; then java_home='/usr/java/jdk1.5' java_home_Linux_x86_64='/usr/java/jdk1.5' java_home_dev='/usr/java/jdk1.6_x64' else # Optionally specify java path for
```

```
all available OS / architecture combinations java_home_Linux="${ij_path}/jre"
java_home_Linux_x86_64="${ij_path}/jre64" java_home_SunOS="${ij_path}/jre64"
java_home_Darwin="" # fi ijadmin='j.jackson AT ion.ucl.ac.uk' # DOCUMENTATION
URL #doc_url='http://rsb.info.nih.gov/ij/'
doc_url='http://saturn/~jjackson/imagej' # TEMP FOLDER ij_tmp='/tmp/imagej' #
LOG FILE ij_log="${ij_tmp}/log.txt" # default behaviour when an ImageJ window
is already open newwindow='true' #newwindow='false' # macro argument
delimiter character separator=':' # a ' ' may work provided no arguments
would contain spaces # use macro functions: args=getArgument();
argArray=split(args, ':'); # to recover macro arguments ############ DEFAULT
MEMORY SETTINGS ########################## declare -i default_mem=900 declare
-i min_mem=32 declare -i max_32bit=1800 # empirical declare -i
max_64bit=17000000000 # conservative ############ SITE SPECIFIC MODULES
########################## # JAR libraries for additional modules may be
placed in ${ij_path}/lib # jmf.jar jai_codec.jar jai_core.jar
mlibwrapper_jai.jar # Native libraries may be placed in ${ij_path}/lib/$OS #
where OS matches the output of the 'uname' command ############ END SITE
SPECIFIC VARIABLES ########################## OS=$(uname) processor=$(uname -m)
# -p doesn't work on ubuntu declare -i mem declare -i max_mem declare -i
free_mem java_home="${java_home:-$JAVA_HOME}" if [[ "$OS" == 'SunOS' ]] ;
then java_arch='-d64' JAVA_HOME="${java_home_SunOS:-$java_home}"
max_mem=`vmstat | awk 'BEGIN{maxMem='$max_64bit'} NR == 3 {fmem=int($5 /
1024); if (fmem < maxMem) {print fmem} else {print maxMem}}'`
free_mem="max_mem" mem=${free_mem}/2 elif [[ "$OS" == 'Linux' ]] ; then if [[
"$processor" == 'x86_64' ]] ; then java_arch='-d64'
JAVA_HOME="${java_home_Linux_x86_64:-$java_home}" max_mem=`free | awk -v
maxMem=$max_64bit 'NR == 2 {fmem=int($2 / 1024); if (fmem < maxMem) {print
fmem} else {print maxMem}}'` free_mem=`free | awk -v maxMem=$max_64bit 'NR ==
3 {fmem=int($4 / 1024); if (fmem < maxMem) {print fmem} else {print
maxMem}}'` else java_arch='-d32' JAVA_HOME="${java_home_Linux:-$java_home}"
max_mem=`free | awk -v maxMem=$max_32bit 'NR == 2 {fmem=int($2 / 1024); if
(fmem < maxMem) {print fmem} else {print maxMem}}'` free_mem=`free | awk -v
maxMem=$max_32bit 'NR == 3 {fmem=int($4 / 1024); if (fmem < maxMem) {print
fmem} else {print maxMem}}'` fi elif [[ "$OS" == Darwin ]] ; then
java_arch='-d64' JAVA_HOME="${java_home_Darwin:-$java_home}" max_mem=`vm_stat
| awk 'NR == 2 {free=$3} NR == 3 {active=$3} NR == 4 {inactive=$3} END{print
int((free+active+inactive)/256)}'` free_mem=`vm_stat | awk
'BEGIN{maxMem='$max_64bit'} NR == 2 {fmem=int($3 / 256)} NR == 4 {imem=int($3
/ 256)} END{amem=fmem+imem ; if (amem < maxMem) {print amem} else {print
maxMem}}'` fi mem=${free_mem}/3*2 (( $mem > $default_mem || $mem < $min_mem
)) && mem=$default_mem if [[ -f ${JAVA_HOME}/bin/java ]] ; then
JAVA=${JAVA_HOME}/bin/java else JAVA=${JAVA_HOME}/java unset JAVA_HOME #
confuses Mac java fi # if tools.jar is not in ${ij_path}/jre/lib/ext/ edit
the 'tools=' line # to point to tools.jar. The -compile switch will load
tools.jar into the # classpath and enable plugins to be compiled in imagej if
[[ -f "${ij_path}/tools.jar" ]] ; then tools="${ij_path}/tools.jar" else
tools='' fi # End Site specific variables
---------------------------------------------------------- # other variables
dir=`pwd` user=`whoami` host=`hostname` # `hostname -s` is better but not
solaris 8 compat. if [[ -z "$DISPLAY" ]] ; then echo 'Display variable not
set' echo 'If ImageJ fails to load, try ' echo '% setenv DISPLAY
```

```
yourcomputer:0' echo 'if you use the "csh" or for "bash" try' echo '% export
DISPLAY=yourcomputer:0' display='default' else display="${DISPLAY##*/}" #
display variable on Darwin can look like this: '/tmp/launch-si9S06/:0' fi
declare -i port=0 declare -i verbosity=0 declare -i max_attempts=10 images=''
macrocmd='' macroargs='' function usage { echo echo 'Image display and
analysis program. Opens formats including:' echo 'UNC, Analyze, Dicom, NIFTI,
Tiff, Jpeg, Gif, PNG ...' echo echo 'imagej [options] image [ image2 ...
image3 ]' echo ' -h print help and more options' echo ' -o open images in an
open ImageJ panel' echo ' -p <N> open images in ImageJ panel number <N>' echo
" -x <MB> set available memory (default=${mem} max=${max_mem})" echo }
function fullusage { echo echo 'Image display and analysis program. Opens
formats including:' echo 'UNC, Analyze, Dicom, NIFTI, Tiff, Jpeg, Gif, PNG
...' echo echo 'imagej [options] image [ image2 ... image3 ] -> open images'
echo echo 'basic options:' echo ' -h print help and more options' echo ' -o
open images in existing ImageJ panel if one exists' echo ' -p <N> open images
in existing ImageJ panel number <N>' echo " -x <MB> set available memory
(default=${mem} max=${max_mem})" echo echo 'advanced options:' echo ' -c
enable plugin compilation within imagej' echo ' -d use development version'
echo " -j use development java version ($java_home_dev)" echo ' -v be verbose
(vv or vvv increases verbosity)' echo echo 'options for batch processing:'
echo " -e 'Macro Code' execute macro code" echo " -r 'Menu Command' run menu
command" echo "Quotation marks '' are required around commands including
spaces" echo 'Commands can be sent to open ImageJ panels with the -p option'
echo echo 'options for macros:' echo 'imagej [-i image] [-b|-m] [arg1 ...
argN] ' echo ' -b macro run macro without graphics window' echo ' -m macro
run macro' echo '"image" will be opened before macro is run' echo 'all
following arguments are passed to macro' echo echo "Documentation - $doc_url
" echo "Report problems with this software to $ijadmin" echo } function
macroCmdError { fullusage echo 'Only one command option (-b -e -m OR -r) may
be specified' 1>&2 echo "Macro commands can't be used with the '-o' option"
1>&2 exit 1 } # parse input arguments while getopts b:cde:hi:jm:nop:r:vx:
options do case $options in b) [[ -n "$macrocmd" ]] && macroCmdError # exits
macrocmd="-batch ${OPTARG}" display='batch' newwindow='false' ;; c)
modules="${modules:-}${modules+:}${tools}" ;; d) ij_path="$ij_path_dev" ;; e)
[[ -n "$macrocmd" ]] && macroCmdError # exits macrocmd='-eval'
macroargs="'${OPTARG}'" ;; h) fullusage exit 0 ;; i)
images="${images}'${OPTARG}' " ;; j) JAVA_HOME="$java_home_dev" ;; m) [[ -n
"$macrocmd" ]] && macroCmdError # exits macrocmd="-macro ${OPTARG}" ;; n)
newwindow='true' ;; o) [[ -n "$macrocmd" ]] && macroCmdError # exits
newwindow='false' ;; p) newwindow='false' port="${OPTARG}" if (( "$port" < 1
|| "$port" > 99 )) ; then echo "${OPTARG} is not a permissible value for port
number (-p)" 1>&2 exit 1 fi ;; r) [[ -n "$macrocmd" ]] && macroCmdError #
exits macrocmd='-run' macroargs="'${OPTARG}'" ;; v) verbosity=verbosity+1 if
(( $verbosity == 2 )) ; then set -x ; fi if (( $verbosity == 3 )) ; then set
-v ; fi ;; x) mem="${OPTARG}" newwindow='true' if (( $mem < $min_mem || $mem
> $max_mem )) ; then echo "${OPTARG} is not a permissible value for memory
(-x)" 1>&2 echo "min=${min_mem}, max=${max_mem}" 1>&2 exit 1 fi ;; \?) usage
exit 1 ;; esac done declare -i i=1 while (( i < $OPTIND )) ; do shift i=i+1
done if [[ "$#" == 0 && -z "$macrocmd" ]] ; then usage fi # The best way to
install .jar libraries required by plugins is to copy them # to the imagej
plugins/jars directory # Alternatively, either copy them to
```

```
${ij_path}/jre/lib/ext/ or add the .jar # filepath to the modules line below. Paths are separated by a colon # Classpath must follow command line arguments, as ij_path is dependent on the -d option # Resolving ij.jar path. If ij.jar is a symbolic link to ij_<version>.jar # this allows updating ij.jar without crashing running sessions ij_jar_path=$(ReadLink ${ij_path}/ij.jar) for mod_jar in ${ij_path}/lib/*jar ; do modules="${modules:-}${modules+:}$mod_jar" done modules="-cp ${ij_jar_path}:${modules+:}${modules:-}" #${ij_path}/plugins/jars/dcmie.jar export LD_LIBRARY_PATH="${ij_path}/lib/${OS}_$processor" if (( $verbosity > 0 )) ; then echo "LD_LIBRARY_PATH=$LD_LIBRARY_PATH" fi # -b and -m options only: # remaining command line arguments are passed as macro arguments # separated by $separator if [[ -n "$macrocmd" && -z "$macroargs" ]] ; then while (( "$#" > 0 )) ; do if [[ -z "$macroargs" ]] ; then macroargs="${1}" else macroargs="${macroargs}${separator}${1}" fi shift done macroargs="'$macroargs'" # if user hasn't requested a specific port number # and it's not a batch mode macro, set display to 'macro' # to prevent other instances accessing this ImageJ window if [[ "$display" != "batch" && "$port" == 0 ]] ; then display="macro" newwindow='true' # should be redundant fi fi # PROTECT POSSIBLE SPACES IN IMAGE FILENAMES if (( "$#" > 0 )) ; then while (( "$#" > 0 )) ; do filearg="${1}" if [[ ! -f "$filearg" ]] ; then echo "Warning: image '$filearg' may not exist" >&2 fi # full file path required when sending images to running ImageJ panel if [[ "${newwindow}" == 'false' && -f "${filearg}" ]] && ! expr "${filearg}" : '/.*' > /dev/null; then filearg="$(ReadLink ${filearg})" fi images="${images}'$filearg' " shift done fi # CREATE IMAGEJ SOCKET-LOCK DIRECTORY IF NON EXISTANT if [[ ! -d "$ij_tmp" ]] ; then mkdir "$ij_tmp" chmod 777 "$ij_tmp" fi # CREATE IMAGEJ LOG FILE IF NON EXISTANT if [[ -n "$ij_log" && ! -f "$ij_log" ]] ; then touch "$ij_log" chmod 666 "$ij_log" fi # CREATES A TEMP FILE INDICATING A PORT IS IN USE BY IMAGEJ cd "$ij_tmp" declare -i count=1 portopen='false' lockFileCreated='false' declare -a locklist=(`ls | grep '[0-9][0-9]-.*'`) [[ -n "$ij_log" ]] && echo -e "$$\t$(date)\tNew Window = $newwindow" >> "$ij_log" 2> /dev/null [[ -n "$ij_log" ]] && echo -e "$$\t$(date)\tPort = $port" >> "$ij_log" 2> /dev/null [[ -n "$ij_log" ]] && echo -e "$$\t$(date)\tlocklist: \n ${locklist[*]}" >> "$ij_log" 2> /dev/null if (( $verbosity > 0 )) ; then echo -e "locklist: \n ${locklist[*]}" ; fi # PORT SPECIFIED BY USER if (( $port > 0 )) ; then # look for a lock on the port specified for lockname in ${locklist[*]} ; do prefix=`printf '%02u' $port` if [[ "$lockname" == ${prefix}-${user}-${host}* ]] ; then # found lock on the requested port, owned by user on current display portopen='true' if (( $verbosity > 0 )) ; then echo "Using socket lock: $lockname" ; fi count=$port break elif [[ "$lockname" == ${prefix}-* ]] ; then echo "Port $port is in use by some other user or a different host" 1>&2 if (( $verbosity > 0 )) ; then echo "Port lock: $lockname" ; fi exit 1 fi done # specified port not in use count=$port # IF EXISTING WINDOW IS REQUESTED, LOOK FOR LISTENING PORT elif [[ "$newwindow" == 'false' && ${#locklist} != 0 ]] ; then # look for a lock on the current display for this user for lockname in ${locklist[*]} ; do if [[ "$lockname" == [0-9][0-9]-${user}-${host}-${display} ]] ; then portopen='true' if (( $verbosity > 0 )) ; then echo "Found socket lock: $lockname" ; fi # if a matching user/display is found, use this one count="${lockname%-*-*-*}" #count=`echo $lockname | sed -e 's/^\([0-9][0-9]\).*/\1/' -e 's/^0//'` # csh? break fi done fi # IF A NEW PORT IS TO BE USED declare -i attempts=0 while [[
```

```
"$portopen" == 'false' ]] ; do # new window requested or no matching port found # if port is not specified, look for first free port if (( "$port" == 0 )) ; then if (( ${#locklist} == 0 )) ; then # no active locks - use first port count=1 else # active locks - check each port number so see if it is in use # this is not synchronised!! count=0 inuse='true' while [[ "$inuse" == 'true' ]] ; do count=count+1 prefix=`printf '%02u' $count` inuse='false' for lockname in ${locklist[*]} ; do if [[ "$lockname" == ${prefix}-* ]] ; then inuse='true' fi done done fi fi # CREATING A NEW PORT LOCK prefix=`printf '%02u' $count` lockname=${prefix}-${user}-${host}-${display} [[ -n "$ij_log" ]] && echo -e "$$\t$(date)\tCreating lock\t$lockname" >> "$ij_log" 2> /dev/null (( $verbosity > 0 )) && echo -n "creating lock $lockname ... " echo $$ > $lockname 2> /dev/null if [[ "$(head -n 1 $lockname 2> /dev/null)" == $$ ]] ; then # port lock successful trap '\rm -f ${ij_tmp}/$lockname >/dev/null ; [[ -n "$ij_log" ]] && echo -e "$$\t$(date)\tReleasing lock\t$lockname" >> "$ij_log" 2> /dev/null' EXIT TERM KILL # Quitting ImageJ sends EXIT, as does a kill/kill -9 # CTRL+C in terminal sends INT + EXIT # System shutdown sends TERM (+EXIT??) portopen='true' if (( $verbosity > 0 )) ; then echo 'done' ; fi lockFileCreated='true' if [[ -z "$macrocmd" ]] ; then echo 'Open other images in this ImageJ panel as follows:' echo " imagej -p $count <image1> [<image2> ... <imageN>]" fi [[ -n "$ij_log" ]] && echo -e "$$\t$(date)\tSocket lock:\t$lockname" >> "$ij_log" 2> /dev/null if (( $verbosity > 0 )) ; then echo "Socket lock: $lockname" ; fi echo else # port lock unsuccessful (simultaneous instances?) [[ -n "$ij_log" ]] && echo -e "$$\t$(date)\tAttempted lock failed\t$lockname" >> "$ij_log" 2> /dev/null if (( $attempts > $max_attempts )) ; then echo "Failed to secure a port lock for ImageJ after $max_attempts" >&2 [[ -n "$ij_log" ]] && echo -e "$$\t$(date)\tGiving up\t$lockname" >> "$ij_log" 2> /dev/null fi # REREAD LOCK LIST declare -a locklist=(`ls | grep '[0-9][0-9]-.*'`) fi attempts=$attempts+1 done if (( $verbosity > 0 )) ; then echo ${JAVA} ${java_arch} -mx${mem}m ${modules} ij.ImageJ -ijpath ${ij_path} -port${count} ${images} ${macrocmd} ${macroargs} fi cd "$dir" eval "${JAVA} ${java_arch} -mx${mem}m ${modules} ij.ImageJ -ijpath ${ij_path} -port${count} ${images} ${macrocmd} ${macroargs} " exit 0
```

```
))&amp;quot;
# IMAGEJ PATH - development installation (optional)
ij_path_dev=&amp;#039;/Users/jjackson/ImageJ&amp;#039;
# JAVA PATH
# assumes executable is ${java_home}/bin/java
# set java_home variables =&amp;#039;&amp;#039; to use JAVA_HOME environment variable
# Get a sensible default
if which java &amp;gt; /dev/null ; then
java_home=&amp;quot;$(dirname $(ReadLink $(which java)))&amp;quot;
fi
# The following allow platform specific defaults for the java path
# these take preference over the default if set
if [[ -d /usr/java/jdk1.6_x64 ]] ; then
java_home=&amp;#039;/usr/java/jdk1.5&amp;#039;
java_home_Linux_x86_64=&amp;#039;/usr/java/jdk1.5&amp;#039;
java_home_dev=&amp;#039;/usr/java/jdk1.6_x64&amp;#039;
else
```

```
# Optionally specify java path for all available OS / architecture
combinations
java_home_Linux=&quot;${ij_path}/jre&quot;
java_home_Linux_x86_64=&quot;${ij_path}/jre64&quot;
java_home_SunOS=&quot;${ij_path}/jre64&quot;
java_home_Darwin=&quot;&quot;
#
fi
ijadmin=&#039;j.jackson AT ion.ucl.ac.uk&#039;
# DOCUMENTATION URL
#doc_url=&#039;http://rsb.info.nih.gov/ij/&#039;
doc_url=&#039;http://saturn/~jjackson/imagej&#039;
# TEMP FOLDER
ij_tmp=&#039;/tmp/imagej&#039;
# LOG FILE
ij_log=&quot;${ij_tmp}/log.txt&quot;
# default behaviour when an ImageJ window is already open
newwindow=&#039;true&#039;
#newwindow=&#039;false&#039;
# macro argument delimiter character
separator=&#039;:&#039;
# a &#039; &#039; may work provided no arguments would contain spaces
# use macro functions:  args=getArgument(); argArray=split(args,
&#039;:&#039;);
# to recover macro arguments

############ DEFAULT MEMORY SETTINGS  ##########################

declare -i default_mem=900
declare -i min_mem=32
declare -i max_32bit=1800 # empirical
declare -i max_64bit=17000000000 # conservative

############ SITE SPECIFIC MODULES ##########################

# JAR libraries for additional modules may be placed in ${ij_path}/lib
# jmf.jar jai_codec.jar jai_core.jar mlibwrapper_jai.jar

# Native libraries may be placed in ${ij_path}/lib/$OS
# where OS matches the output of the &#039;uname&#039; command

############ END SITE SPECIFIC VARIABLES ##########################

OS=$(uname)
processor=$(uname -m) # -p doesn&#039;t work on ubuntu
declare -i mem
declare -i max_mem
declare -i free_mem

java_home=&quot;${java_home:-$JAVA_HOME}&quot;
```

```
if [[ &quot;$OS&quot; == &#039;SunOS&#039; ]] ; then
java_arch=&#039;-d64&#039;
JAVA_HOME=&quot;${java_home_SunOS:-$java_home}&quot;
max_mem=`vmstat | awk &#039;BEGIN{maxMem=&#039;$max_64bit&#039;} NR == 3
{fmem=int( / 1024); if (fmem &lt; maxMem) {print fmem} else {print
maxMem}}&#039;`
free_mem=&quot;max_mem&quot;
mem=${free_mem}/2
elif [[ &quot;$OS&quot; == &#039;Linux&#039; ]] ; then
if [[ &quot;$processor&quot; == &#039;x86_64&#039; ]] ; then
java_arch=&#039;-d64&#039;
JAVA_HOME=&quot;${java_home_Linux_x86_64:-$java_home}&quot;
max_mem=`free | awk -v maxMem=$max_64bit &#039;NR == 2 {fmem=int(#!/bin/bash
# A wrapper script to launch imagej from the UNIX command line # Images given
as arguments will be opened, macros may also be given as arguments # Looks
for macros in the imagej macro directory # # This program is free software,
but comes with no warrenty or guarantee # send bug reports or feedback to
jjackson at familyjackson dot net # Author: Jon Jackson # Last modified date:
$Date: 2008-12-04 18:38:47 +0000 (Thu, 04 Dec 2008) $ # $Revision: 64 $ # #
INSTALLATION INSTRUCTIONS # ### WARNING
############################################################## # This file won't
run if edited in 'Notepad' or similar programs # that don't support unix new
line characters
############################################################################ #
Source location: http://rsb.info.nih.gov/ij/download/linux/unix-script.txt #
1) Save this script in the ImageJ directory as 'imagej' # 2) Modify path
variables according to your system # 3) Give the new file execute permission
# 4) Be sure the 'imagej' wrapper is in the 'PATH' ### DEPENDENCIES
##################################################### # readlink: This
script works better with GNU readlink, which is not # installed by default on
MacOS (or Solaris 8) # MacOS X: readlink can be installed with 'MacPorts': #
Install MacPorts and Apple xCode, then run # 'sudo port install coreutils'
and set the PATH variable appropriately # ALL: If no version of readlink is
avaliable, hard code the ij_path and ij_jar_paths # sending images to a
running ImageJ panel may not work (around line 357)
############################################################################ #
setup environment set +u # don't give error for unset variables (matters for
environment variables) shopt -s extglob # allow extended pattern matching
shopt -s expand_aliases # Detect readlink version # if GNU readlink is known
to be installed, this code can be replaced by "alias ReadLink='readlink -f'"
alias ReadLink='readlink' # Default if [[ $(uname) == Darwin ]] && which
greadlink &>/dev/null ; then # Using GNU readlink on MacOS X alias
ReadLink='greadlink -f' elif [[ -f $(which readlink) ]] && readlink --version
| grep coreutils &>/dev/null ; then alias ReadLink='readlink -f' # use GNU
readlink else alias ReadLink='echo' echo "Please install GNU readlink.
Symbolic links may not be resolved properly" >&2 fi ############ SITE
SPECIFIC VARIABLES ########################## # Trailing / is not required for
path variables # IMAGEJ PATH - production installation
#ij_path='/local/ImageJ' ij_path="$(dirname $(ReadLink $0))" # IMAGEJ PATH -
development installation (optional) ij_path_dev='/Users/jjackson/ImageJ' #
JAVA PATH # assumes executable is ${java_home}/bin/java # set java_home
```

```
variables ='' to use JAVA_HOME environment variable # Get a sensible default
if which java > /dev/null ; then java_home="$(dirname $(ReadLink $(which
java)))" fi # The following allow platform specific defaults for the java
path # these take preference over the default if set if [[ -d
/usr/java/jdk1.6_x64 ]] ; then java_home='/usr/java/jdk1.5'
java_home_Linux_x86_64='/usr/java/jdk1.5'
java_home_dev='/usr/java/jdk1.6_x64' else # Optionally specify java path for
all available OS / architecture combinations java_home_Linux="${ij_path}/jre"
java_home_Linux_x86_64="${ij_path}/jre64" java_home_SunOS="${ij_path}/jre64"
java_home_Darwin="" # fi ijadmin='j.jackson AT ion.ucl.ac.uk' # DOCUMENTATION
URL #doc_url='http://rsb.info.nih.gov/ij/'
doc_url='http://saturn/~jjackson/imagej' # TEMP FOLDER ij_tmp='/tmp/imagej' #
LOG FILE ij_log="${ij_tmp}/log.txt" # default behaviour when an ImageJ window
is already open newwindow='true' #newwindow='false' # macro argument
delimiter character separator=':' # a ' ' may work provided no arguments
would contain spaces # use macro functions: args=getArgument();
argArray=split(args, ':'); # to recover macro arguments ############ DEFAULT
MEMORY SETTINGS ########################## declare -i default_mem=900 declare
-i min_mem=32 declare -i max_32bit=1800 # empirical declare -i
max_64bit=17000000000 # conservative ############ SITE SPECIFIC MODULES
########################## # JAR libraries for additional modules may be
placed in ${ij_path}/lib # jmf.jar jai_codec.jar jai_core.jar
mlibwrapper_jai.jar # Native libraries may be placed in ${ij_path}/lib/$OS #
where OS matches the output of the 'uname' command ############ END SITE
SPECIFIC VARIABLES ########################## OS=$(uname) processor=$(uname -m)
# -p doesn't work on ubuntu declare -i mem declare -i max_mem declare -i
free_mem java_home="${java_home:-$JAVA_HOME}" if [[ "$OS" == 'SunOS' ]] ;
then java_arch='-d64' JAVA_HOME="${java_home_SunOS:-$java_home}"
max_mem=`vmstat | awk 'BEGIN{maxMem='$max_64bit'} NR == 3 {fmem=int($5 /
1024); if (fmem < maxMem) {print fmem} else {print maxMem}}'`
free_mem="max_mem" mem=${free_mem}/2 elif [[ "$OS" == 'Linux' ]] ; then if [[
"$processor" == 'x86_64' ]] ; then java_arch='-d64'
JAVA_HOME="${java_home_Linux_x86_64:-$java_home}" max_mem=`free | awk -v
maxMem=$max_64bit 'NR == 2 {fmem=int($2 / 1024); if (fmem < maxMem) {print
fmem} else {print maxMem}}'` free_mem=`free | awk -v maxMem=$max_64bit 'NR ==
3 {fmem=int($4 / 1024); if (fmem < maxMem) {print fmem} else {print
maxMem}}'` else java_arch='-d32' JAVA_HOME="${java_home_Linux:-$java_home}"
max_mem=`free | awk -v maxMem=$max_32bit 'NR == 2 {fmem=int($2 / 1024); if
(fmem < maxMem) {print fmem} else {print maxMem}}'` free_mem=`free | awk -v
maxMem=$max_32bit 'NR == 3 {fmem=int($4 / 1024); if (fmem < maxMem) {print
fmem} else {print maxMem}}'` fi elif [[ "$OS" == Darwin ]] ; then
java_arch='-d64' JAVA_HOME="${java_home_Darwin:-$java_home}" max_mem=`vm_stat
| awk 'NR == 2 {free=$3} NR == 3 {active=$3} NR == 4 {inactive=$3} END{print
int((free+active+inactive)/256)}'` free_mem=`vm_stat | awk
'BEGIN{maxMem='$max_64bit'} NR == 2 {fmem=int($3 / 256)} NR == 4 {imem=int($3
/ 256)} END{amem=fmem+imem ; if (amem < maxMem) {print amem} else {print
maxMem}}'` fi mem=${free_mem}/3*2 (( $mem > $default_mem || $mem < $min_mem
)) && mem=$default_mem if [[ -f ${JAVA_HOME}/bin/java ]] ; then
JAVA=${JAVA_HOME}/bin/java else JAVA=${JAVA_HOME}/java unset JAVA_HOME #
confuses Mac java fi # if tools.jar is not in ${ij_path}/jre/lib/ext/ edit
the 'tools=' line # to point to tools.jar. The -compile switch will load
```

```
tools.jar into the # classpath and enable plugins to be compiled in imagej if
[[ -f "${ij_path}/tools.jar" ]] ; then tools="${ij_path}/tools.jar" else
tools='' fi # End Site specific variables
---------------------------------------------------------- # other variables
dir=`pwd` user=`whoami` host=`hostname` # `hostname -s` is better but not
solaris 8 compat. if [[ -z "$DISPLAY" ]] ; then echo 'Display variable not
set' echo 'If ImageJ fails to load, try ' echo '% setenv DISPLAY
yourcomputer:0' echo 'if you use the "csh" or for "bash" try' echo '% export
DISPLAY=yourcomputer:0' display='default' else display="${DISPLAY##*/}" #
display variable on Darwin can look like this: '/tmp/launch-si9S06/:0' fi
declare -i port=0 declare -i verbosity=0 declare -i max_attempts=10 images=''
macrocmd='' macroargs='' function usage { echo echo 'Image display and
analysis program. Opens formats including:' echo 'UNC, Analyze, Dicom, NIFTI,
Tiff, Jpeg, Gif, PNG ...' echo echo 'imagej [options] image [ image2 ...
image3 ]' echo ' -h print help and more options' echo ' -o open images in an
open ImageJ panel' echo ' -p <N> open images in ImageJ panel number <N>' echo
" -x <MB> set available memory (default=${mem} max=${max_mem})" echo }
function fullusage { echo echo 'Image display and analysis program. Opens
formats including:' echo 'UNC, Analyze, Dicom, NIFTI, Tiff, Jpeg, Gif, PNG
...' echo echo 'imagej [options] image [ image2 ... image3 ] -> open images'
echo echo 'basic options:' echo ' -h print help and more options' echo ' -o
open images in existing ImageJ panel if one exists' echo ' -p <N> open images
in existing ImageJ panel number <N>' echo " -x <MB> set available memory
(default=${mem} max=${max_mem})" echo echo 'advanced options:' echo ' -c
enable plugin compilation within imagej' echo ' -d use development version'
echo " -j use development java version ($java_home_dev)" echo ' -v be verbose
(vv or vvv increases verbosity)' echo echo 'options for batch processing:'
echo " -e 'Macro Code' execute macro code" echo " -r 'Menu Command' run menu
command" echo "Quotation marks '' are required around commands including
spaces" echo 'Commands can be sent to open ImageJ panels with the -p option'
echo echo 'options for macros:' echo 'imagej [-i image] [-b|-m] [arg1 ...
argN] ' echo ' -b macro run macro without graphics window' echo ' -m macro
run macro' echo '"image" will be opened before macro is run' echo 'all
following arguments are passed to macro' echo echo "Documentation - $doc_url
" echo "Report problems with this software to $ijadmin" echo } function
macroCmdError { fullusage echo 'Only one command option (-b -e -m OR -r) may
be specified' 1>&2 echo "Macro commands can't be used with the '-o' option"
1>&2 exit 1 } # parse input arguments while getopts b:cde:hi:jm:nop:r:vx:
options do case $options in b) [[ -n "$macrocmd" ]] && macroCmdError # exits
macrocmd="-batch ${OPTARG}" display='batch' newwindow='false' ;; c)
modules="${modules:-}${modules+:}${tools}" ;; d) ij_path="$ij_path_dev" ;; e)
[[ -n "$macrocmd" ]] && macroCmdError # exits macrocmd='-eval'
macroargs="'${OPTARG}'" ;; h) fullusage exit 0 ;; i)
images="${images}'${OPTARG}' " ;; j) JAVA_HOME="$java_home_dev" ;; m) [[ -n
"$macrocmd" ]] && macroCmdError # exits macrocmd="-macro ${OPTARG}" ;; n)
newwindow='true' ;; o) [[ -n "$macrocmd" ]] && macroCmdError # exits
newwindow='false' ;; p) newwindow='false' port="${OPTARG}" if (( "$port" < 1
|| "$port" > 99 )) ; then echo "${OPTARG} is not a permissible value for port
number (-p)" 1>&2 exit 1 fi ;; r) [[ -n "$macrocmd" ]] && macroCmdError #
exits macrocmd='-run' macroargs="'${OPTARG}'" ;; v) verbosity=verbosity+1 if
(( $verbosity == 2 )) ; then set -x ; fi if (( $verbosity == 3 )) ; then set
```

```
-v ; fi ;; x) mem="${OPTARG}" newwindow='true' if (( $mem < $min_mem || $mem
> $max_mem )) ; then echo "${OPTARG} is not a permissible value for memory
(-x)" 1>&2 echo "min=${min_mem}, max=${max_mem}" 1>&2 exit 1 fi ;; \?) usage
exit 1 ;; esac done declare -i i=1 while (( i < $OPTIND )) ; do shift i=i+1
done if [[ "$#" == 0 && -z "$macrocmd" ]] ; then usage fi # The best way to
install .jar libraries required by plugins is to copy them # to the imagej
plugins/jars directory # Alternatively, either copy them to
${ij_path}/jre/lib/ext/ or add the .jar # filepath to the modules line below.
Paths are separated by a colon # Classpath must follow command line arguments,
as ij_path is dependent on the -d option # Resolving ij.jar path. If ij.jar
is a symbolic link to ij_<version>.jar # this allows updating ij.jar without
crashing running sessions ij_jar_path=$(ReadLink ${ij_path}/ij.jar) for
mod_jar in ${ij_path}/lib/*jar ; do modules="${modules:-}${modules+:}$mod_jar"
done modules="-cp ${ij_jar_path}:${modules+:}${modules:-}"
#${ij_path}/plugins/jars/dcmie.jar export
LD_LIBRARY_PATH="${ij_path}/lib/${OS}_$processor" if (( $verbosity > 0 )) ;
then echo "LD_LIBRARY_PATH=$LD_LIBRARY_PATH" fi # -b and -m options only: #
remaining command line arguments are passed as macro arguments # separated by
$separator if [[ -n "$macrocmd" && -z "$macroargs" ]] ; then while (( "$#" >
0 )) ; do if [[ -z "$macroargs" ]] ; then macroargs="${1}" else
macroargs="${macroargs}${separator}${1}" fi shift done
macroargs="'$macroargs'" # if user hasn't requested a specific port number #
and it's not a batch mode macro, set display to 'macro' # to prevent other
instances accessing this ImageJ window if [[ "$display" != "batch" && "$port"
== 0 ]] ; then display="macro" newwindow='true' # should be redundant fi fi #
PROTECT POSSIBLE SPACES IN IMAGE FILENAMES if (( "$#" > 0 )) ; then while ((
"$#" > 0 )) ; do filearg="${1}" if [[ ! -f "$filearg" ]] ; then echo "Warning:
image '$filearg' may not exist" >&2 fi # full file path required when sending
images to running ImageJ panel if [[ "${newwindow}" == 'false' && -f
"${filearg}" ]] && ! expr "${filearg}" : '/.*' > /dev/null; then
filearg="$(ReadLink ${filearg})" fi images="${images}'$filearg' " shift done
fi # CREATE IMAGEJ SOCKET-LOCK DIRECTORY IF NON EXISTANT if [[ ! -d "$ij_tmp"
]] ; then mkdir "$ij_tmp" chmod 777 "$ij_tmp" fi # CREATE IMAGEJ LOG FILE IF
NON EXISTANT if [[ -n "$ij_log" && ! -f "$ij_log" ]] ; then touch "$ij_log"
chmod 666 "$ij_log" fi # CREATES A TEMP FILE INDICATING A PORT IS IN USE BY
IMAGEJ cd "$ij_tmp" declare -i count=1 portopen='false'
lockFileCreated='false' declare -a locklist=(`ls | grep '[0-9][0-9]-.*'`) [[
-n "$ij_log" ]] && echo -e "$$\t$(date)\tNew Window = $newwindow" >> "$ij_log"
2> /dev/null [[ -n "$ij_log" ]] && echo -e "$$\t$(date)\tPort = $port" >>
"$ij_log" 2> /dev/null [[ -n "$ij_log" ]] && echo -e "$$\t$(date)\tlocklist:
\n ${locklist[*]}" >> "$ij_log" 2> /dev/null if (( $verbosity > 0 )) ; then
echo -e "locklist: \n ${locklist[*]}" ; fi # PORT SPECIFIED BY USER if ((
$port > 0 )) ; then # look for a lock on the port specified for lockname in
${locklist[*]} ; do prefix=`printf '%02u' $port` if [[ "$lockname" ==
${prefix}-${user}-${host}* ]] ; then # found lock on the requested port,
owned by user on current display portopen='true' if (( $verbosity > 0 )) ;
then echo "Using socket lock: $lockname" ; fi count=$port break elif [[
"$lockname" == ${prefix}-* ]] ; then echo "Port $port is in use by some other
user or a different host" 1>&2 if (( $verbosity > 0 )) ; then echo "Port lock:
$lockname" ; fi exit 1 fi done # specified port not in use count=$port # IF
EXISTING WINDOW IS REQUESTED, LOOK FOR LISTENING PORT elif [[ "$newwindow" ==
```

```
'false' && ${#locklist} != 0 ]] ; then # look for a lock on the current display for this user for lockname in ${locklist[*]} ; do if [[ "$lockname" == [0-9][0-9]-${user}-${host}-${display} ]] ; then portopen='true' if (( $verbosity > 0 )) ; then echo "Found socket lock: $lockname" ; fi # if a matching user/display is found, use this one count="${lockname%-*-*-*}" #count=`echo $lockname | sed -e 's/^\([0-9][0-9]\).*/\1/' -e 's/^0//'` # csh? break fi done fi # IF A NEW PORT IS TO BE USED declare -i attempts=0 while [[ "$portopen" == 'false' ]] ; do # new window requested or no matching port found # if port is not specified, look for first free port if (( "$port" == 0 )) ; then if (( ${#locklist} == 0 )) ; then # no active locks - use first port count=1 else # active locks - check each port number so see if it is in use # this is not synchronised!! count=0 inuse='true' while [[ "$inuse" == 'true' ]] ; do count=count+1 prefix=`printf '%02u' $count` inuse='false' for lockname in ${locklist[*]} ; do if [[ "$lockname" == ${prefix}-* ]] ; then inuse='true' fi done done fi fi # CREATING A NEW PORT LOCK prefix=`printf '%02u' $count` lockname=${prefix}-${user}-${host}-${display} [[ -n "$ij_log" ]] && echo -e "$$\t$(date)\tCreating lock\t$lockname" >> "$ij_log" 2> /dev/null (( $verbosity > 0 )) && echo -n "creating lock $lockname ... " echo $$ > $lockname 2> /dev/null if [[ "$(head -n 1 $lockname 2> /dev/null)" == $$ ]] ; then # port lock successful trap '\rm -f ${ij_tmp}/$lockname >/dev/null ; [[ -n "$ij_log" ]] && echo -e "$$\t$(date)\tReleasing lock\t$lockname" >> "$ij_log" 2> /dev/null' EXIT TERM KILL # Quitting ImageJ sends EXIT, as does a kill/kill -9 # CTRL+C in terminal sends INT + EXIT # System shutdown sends TERM (+EXIT??) portopen='true' if (( $verbosity > 0 )) ; then echo 'done' ; fi lockFileCreated='true' if [[ -z "$macrocmd" ]] ; then echo 'Open other images in this ImageJ panel as follows:' echo " imagej -p $count <image1> [<image2> ... <imageN>]" fi [[ -n "$ij_log" ]] && echo -e "$$\t$(date)\tSocket lock:\t$lockname" >> "$ij_log" 2> /dev/null if (( $verbosity > 0 )) ; then echo "Socket lock: $lockname" ; fi echo else # port lock unsuccessful (simultaneous instances?) [[ -n "$ij_log" ]] && echo -e "$$\t$(date)\tAttempted lock failed\t$lockname" >> "$ij_log" 2> /dev/null if (( $attempts > $max_attempts )) ; then echo "Failed to secure a port lock for ImageJ after $max_attempts" >&2 [[ -n "$ij_log" ]] && echo -e "$$\t$(date)\tGiving up\t$lockname" >> "$ij_log" 2> /dev/null fi # REREAD LOCK LIST declare -a locklist=(`ls | grep '[0-9][0-9]-.*'`) fi attempts=$attempts+1 done if (( $verbosity > 0 )) ; then echo ${JAVA} ${java_arch} -mx${mem}m ${modules} ij.ImageJ -ijpath ${ij_path} -port${count} ${images} ${macrocmd} ${macroargs} fi cd "$dir" eval "${JAVA} ${java_arch} -mx${mem}m ${modules} ij.ImageJ -ijpath ${ij_path} -port${count} ${images} ${macrocmd} ${macroargs} " exit 0  / 1024); if (fmem &lt; maxMem) {print fmem} else {print maxMem}}&#039;`
free_mem=`free | awk -v maxMem=$max_64bit &#039;NR == 3 {fmem=int( / 1024); if (fmem &lt; maxMem) {print fmem} else {print maxMem}}&#039;`
else
java_arch=&#039;-d32&#039;
JAVA_HOME=&quot;${java_home_Linux:-$java_home}&quot;
max_mem=`free | awk -v maxMem=$max_32bit &#039;NR == 2 {fmem=int(#!/bin/bash # A wrapper script to launch imagej from the UNIX command line # Images given as arguments will be opened, macros may also be given as arguments # Looks for macros in the imagej macro directory # # This program is free software, but comes with no warrenty or guarantee # send bug reports or feedback to
```

```
jjackson at familyjackson dot net # Author: Jon Jackson # Last modified date: $Date: 2008-12-04 18:38:47 +0000 (Thu, 04 Dec 2008) $ # $Revision: 64 $ # # INSTALLATION INSTRUCTIONS # ### WARNING ############################################################ # This file won't run if edited in 'Notepad' or similar programs # that don't support unix new line characters ######################################################################## # Source location: http://rsb.info.nih.gov/ij/download/linux/unix-script.txt # 1) Save this script in the ImageJ directory as 'imagej' # 2) Modify path variables according to your system # 3) Give the new file execute permission # 4) Be sure the 'imagej' wrapper is in the 'PATH' ### DEPENDENCIES ##################################################### # readlink: This script works better with GNU readlink, which is not # installed by default on MacOS (or Solaris 8) # MacOS X: readlink can be installed with 'MacPorts': # Install MacPorts and Apple xCode, then run # 'sudo port install coreutils' and set the PATH variable appropriately # ALL: If no version of readlink is avaliable, hard code the ij_path and ij_jar_paths # sending images to a running ImageJ panel may not work (around line 357) ######################################################################## # setup environment set +u # don't give error for unset variables (matters for environment variables) shopt -s extglob # allow extended pattern matching shopt -s expand_aliases # Detect readlink version # if GNU readlink is known to be installed, this code can be replaced by "alias ReadLink='readlink -f'" alias ReadLink='readlink' # Default if [[ $(uname) == Darwin ]] && which greadlink &>/dev/null ; then # Using GNU readlink on MacOS X alias ReadLink='greadlink -f' elif [[ -f $(which readlink) ]] && readlink --version | grep coreutils &>/dev/null ; then alias ReadLink='readlink -f' # use GNU readlink else alias ReadLink='echo' echo "Please install GNU readlink. Symbolic links may not be resolved properly" >&2 fi ############ SITE SPECIFIC VARIABLES ########################## # Trailing / is not required for path variables # IMAGEJ PATH - production installation #ij_path='/local/ImageJ' ij_path="$(dirname $(ReadLink $0))" # IMAGEJ PATH - development installation (optional) ij_path_dev='/Users/jjackson/ImageJ' # JAVA PATH # assumes executable is ${java_home}/bin/java # set java_home variables ='' to use JAVA_HOME environment variable # Get a sensible default if which java > /dev/null ; then java_home="$(dirname $(ReadLink $(which java)))" fi # The following allow platform specific defaults for the java path # these take preference over the default if set if [[ -d /usr/java/jdk1.6_x64 ]] ; then java_home='/usr/java/jdk1.5' java_home_Linux_x86_64='/usr/java/jdk1.5' java_home_dev='/usr/java/jdk1.6_x64' else # Optionally specify java path for all available OS / architecture combinations java_home_Linux="${ij_path}/jre" java_home_Linux_x86_64="${ij_path}/jre64" java_home_SunOS="${ij_path}/jre64" java_home_Darwin="" # fi ijadmin='j.jackson AT ion.ucl.ac.uk' # DOCUMENTATION URL #doc_url='http://rsb.info.nih.gov/ij/' doc_url='http://saturn/~jjackson/imagej' # TEMP FOLDER ij_tmp='/tmp/imagej' # LOG FILE ij_log="${ij_tmp}/log.txt" # default behaviour when an ImageJ window is already open newwindow='true' #newwindow='false' # macro argument delimiter character separator=':' # a ' ' may work provided no arguments would contain spaces # use macro functions: args=getArgument(); argArray=split(args, ':'); # to recover macro arguments ############ DEFAULT
```

```
MEMORY SETTINGS ########################## declare -i default_mem=900 declare
-i min_mem=32 declare -i max_32bit=1800 # empirical declare -i
max_64bit=17000000000 # conservative ############ SITE SPECIFIC MODULES
########################## # JAR libraries for additional modules may be
placed in ${ij_path}/lib # jmf.jar jai_codec.jar jai_core.jar
mlibwrapper_jai.jar # Native libraries may be placed in ${ij_path}/lib/$OS #
where OS matches the output of the 'uname' command ############ END SITE
SPECIFIC VARIABLES ########################## OS=$(uname) processor=$(uname -m)
# -p doesn't work on ubuntu declare -i mem declare -i max_mem declare -i
free_mem java_home="${java_home:-$JAVA_HOME}" if [[ "$OS" == 'SunOS' ]] ;
then java_arch='-d64' JAVA_HOME="${java_home_SunOS:-$java_home}"
max_mem=`vmstat | awk 'BEGIN{maxMem='$max_64bit'} NR == 3 {fmem=int($5 /
1024); if (fmem < maxMem) {print fmem} else {print maxMem}}'`
free_mem="max_mem" mem=${free_mem}/2 elif [[ "$OS" == 'Linux' ]] ; then if [[
"$processor" == 'x86_64' ]] ; then java_arch='-d64'
JAVA_HOME="${java_home_Linux_x86_64:-$java_home}" max_mem=`free | awk -v
maxMem=$max_64bit 'NR == 2 {fmem=int($2 / 1024); if (fmem < maxMem) {print
fmem} else {print maxMem}}'` free_mem=`free | awk -v maxMem=$max_64bit 'NR ==
3 {fmem=int($4 / 1024); if (fmem < maxMem) {print fmem} else {print
maxMem}}'` else java_arch='-d32' JAVA_HOME="${java_home_Linux:-$java_home}"
max_mem=`free | awk -v maxMem=$max_32bit 'NR == 2 {fmem=int($2 / 1024); if
(fmem < maxMem) {print fmem} else {print maxMem}}'` free_mem=`free | awk -v
maxMem=$max_32bit 'NR == 3 {fmem=int($4 / 1024); if (fmem < maxMem) {print
fmem} else {print maxMem}}'` fi elif [[ "$OS" == Darwin ]] ; then
java_arch='-d64' JAVA_HOME="${java_home_Darwin:-$java_home}" max_mem=`vm_stat
| awk 'NR == 2 {free=$3} NR == 3 {active=$3} NR == 4 {inactive=$3} END{print
int((free+active+inactive)/256)}'` free_mem=`vm_stat | awk
'BEGIN{maxMem='$max_64bit'} NR == 2 {fmem=int($3 / 256)} NR == 4 {imem=int($3
/ 256)} END{amem=fmem+imem ; if (amem < maxMem) {print amem} else {print
maxMem}}'` fi mem=${free_mem}/3*2 (( $mem > $default_mem || $mem < $min_mem
)) && mem=$default_mem if [[ -f ${JAVA_HOME}/bin/java ]] ; then
JAVA=${JAVA_HOME}/bin/java else JAVA=${JAVA_HOME}/java unset JAVA_HOME #
confuses Mac java fi # if tools.jar is not in ${ij_path}/jre/lib/ext/ edit
the 'tools=' line # to point to tools.jar. The -compile switch will load
tools.jar into the # classpath and enable plugins to be compiled in imagej if
[[ -f "${ij_path}/tools.jar" ]] ; then tools="${ij_path}/tools.jar" else
tools='' fi # End Site specific variables
------------------------------------------------------------ # other variables
dir=`pwd` user=`whoami` host=`hostname` # `hostname -s` is better but not
solaris 8 compat. if [[ -z "$DISPLAY" ]] ; then echo 'Display variable not
set' echo 'If ImageJ fails to load, try ' echo '% setenv DISPLAY
yourcomputer:0' echo 'if you use the "csh" or for "bash" try' echo '% export
DISPLAY=yourcomputer:0' display='default' else display="${DISPLAY##*/}" #
display variable on Darwin can look like this: '/tmp/launch-si9S06/:0' fi
declare -i port=0 declare -i verbosity=0 declare -i max_attempts=10 images=''
macrocmd='' macroargs='' function usage { echo echo 'Image display and
analysis program. Opens formats including:' echo 'UNC, Analyze, Dicom, NIFTI,
Tiff, Jpeg, Gif, PNG ...' echo echo 'imagej [options] image [ image2 ...
image3 ]' echo ' -h print help and more options' echo ' -o open images in an
open ImageJ panel' echo ' -p <N> open images in ImageJ panel number <N>' echo
" -x <MB> set available memory (default=${mem} max=${max_mem})" echo }
```

```
function fullusage { echo echo 'Image display and analysis program. Opens formats including:' echo 'UNC, Analyze, Dicom, NIFTI, Tiff, Jpeg, Gif, PNG ...' echo echo 'imagej [options] image [ image2 ... image3 ] -> open images' echo echo 'basic options:' echo ' -h print help and more options' echo ' -o open images in existing ImageJ panel if one exists' echo ' -p <N> open images in existing ImageJ panel number <N>' echo " -x <MB> set available memory (default=${mem} max=${max_mem})" echo echo 'advanced options:' echo ' -c enable plugin compilation within imagej' echo ' -d use development version' echo " -j use development java version ($java_home_dev)" echo ' -v be verbose (vv or vvv increases verbosity)' echo echo 'options for batch processing:' echo " -e 'Macro Code' execute macro code" echo " -r 'Menu Command' run menu command" echo "Quotation marks '' are required around commands including spaces" echo 'Commands can be sent to open ImageJ panels with the -p option' echo echo 'options for macros:' echo 'imagej [-i image] [-b|-m] [arg1 ... argN] ' echo ' -b macro run macro without graphics window' echo ' -m macro run macro' echo '"image" will be opened before macro is run' echo 'all following arguments are passed to macro' echo echo "Documentation - $doc_url " echo "Report problems with this software to $ijadmin" echo } function macroCmdError { fullusage echo 'Only one command option (-b -e -m OR -r) may be specified' 1>&2 echo "Macro commands can't be used with the '-o' option" 1>&2 exit 1 } # parse input arguments while getopts b:cde:hi:jm:nop:r:vx: options do case $options in b) [[ -n "$macrocmd" ]] && macroCmdError # exits macrocmd="-batch ${OPTARG}" display='batch' newwindow='false' ;; c) modules="${modules:-}${modules+:}${tools}" ;; d) ij_path="$ij_path_dev" ;; e) [[ -n "$macrocmd" ]] && macroCmdError # exits macrocmd='-eval' macroargs="'${OPTARG}'" ;; h) fullusage exit 0 ;; i) images="${images}'${OPTARG}' " ;; j) JAVA_HOME="$java_home_dev" ;; m) [[ -n "$macrocmd" ]] && macroCmdError # exits macrocmd="-macro ${OPTARG}" ;; n) newwindow='true' ;; o) [[ -n "$macrocmd" ]] && macroCmdError # exits newwindow='false' ;; p) newwindow='false' port="${OPTARG}" if (( "$port" < 1 || "$port" > 99 )) ; then echo "${OPTARG} is not a permissible value for port number (-p)" 1>&2 exit 1 fi ;; r) [[ -n "$macrocmd" ]] && macroCmdError # exits macrocmd='-run' macroargs="'${OPTARG}'" ;; v) verbosity=verbosity+1 if (( $verbosity == 2 )) ; then set -x ; fi if (( $verbosity == 3 )) ; then set -v ; fi ;; x) mem="${OPTARG}" newwindow='true' if (( $mem < $min_mem || $mem > $max_mem )) ; then echo "${OPTARG} is not a permissible value for memory (-x)" 1>&2 echo "min=${min_mem}, max=${max_mem}" 1>&2 exit 1 fi ;; \?) usage exit 1 ;; esac done declare -i i=1 while (( i < $OPTIND )) ; do shift i=i+1 done if [[ "$#" == 0 && -z "$macrocmd" ]] ; then usage fi # The best way to install .jar libraries required by plugins is to copy them # to the imagej plugins/jars directory # Alternatively, either copy them to ${ij_path}/jre/lib/ext/ or add the .jar # filepath to the modules line below. Paths are separated by a colon # Classpath must follow command line arguments, as ij_path is dependent on the -d option # Resolving ij.jar path. If ij.jar is a symbolic link to ij_<version>.jar # this allows updating ij.jar without crashing running sessions ij_jar_path=$(ReadLink ${ij_path}/ij.jar) for mod_jar in ${ij_path}/lib/*jar ; do modules="${modules:-}${modules+:}$mod_jar" done modules="-cp ${ij_jar_path}:${modules+:}${modules:-}" #${ij_path}/plugins/jars/dcmie.jar export LD_LIBRARY_PATH="${ij_path}/lib/${OS}_$processor" if (( $verbosity > 0 )) ; then echo "LD_LIBRARY_PATH=$LD_LIBRARY_PATH" fi # -b and -m options only: #
```

```
remaining command line arguments are passed as macro arguments # separated by $separator if [[ -n "$macrocmd" && -z "$macroargs" ]] ; then while (( "$#" > 0 )) ; do if [[ -z "$macroargs" ]] ; then macroargs="${1}" else macroargs="${macroargs}${separator}${1}" fi shift done macroargs="'$macroargs'" # if user hasn't requested a specific port number # and it's not a batch mode macro, set display to 'macro' # to prevent other instances accessing this ImageJ window if [[ "$display" != "batch" && "$port" == 0 ]] ; then display="macro" newwindow='true' # should be redundant fi fi # PROTECT POSSIBLE SPACES IN IMAGE FILENAMES if (( "$#" > 0 )) ; then while (( "$#" > 0 )) ; do filearg="${1}" if [[ ! -f "$filearg" ]] ; then echo "Warning: image '$filearg' may not exist" >&2 fi # full file path required when sending images to running ImageJ panel if [[ "${newwindow}" == 'false' && -f "${filearg}" ]] && ! expr "${filearg}" : '/.*' > /dev/null; then filearg="$(ReadLink ${filearg})" fi images="${images}'$filearg' " shift done fi # CREATE IMAGEJ SOCKET-LOCK DIRECTORY IF NON EXISTANT if [[ ! -d "$ij_tmp" ]] ; then mkdir "$ij_tmp" chmod 777 "$ij_tmp" fi # CREATE IMAGEJ LOG FILE IF NON EXISTANT if [[ -n "$ij_log" && ! -f "$ij_log" ]] ; then touch "$ij_log" chmod 666 "$ij_log" fi # CREATES A TEMP FILE INDICATING A PORT IS IN USE BY IMAGEJ cd "$ij_tmp" declare -i count=1 portopen='false' lockFileCreated='false' declare -a locklist=(`ls | grep '[0-9][0-9]-.*'`) [[ -n "$ij_log" ]] && echo -e "$$\t$(date)\tNew Window = $newwindow" >> "$ij_log" 2> /dev/null [[ -n "$ij_log" ]] && echo -e "$$\t$(date)\tPort = $port" >> "$ij_log" 2> /dev/null [[ -n "$ij_log" ]] && echo -e "$$\t$(date)\tlocklist: \n ${locklist[*]}" >> "$ij_log" 2> /dev/null if (( $verbosity > 0 )) ; then echo -e "locklist: \n ${locklist[*]}" ; fi # PORT SPECIFIED BY USER if (( $port > 0 )) ; then # look for a lock on the port specified for lockname in ${locklist[*]} ; do prefix=`printf '%02u' $port` if [[ "$lockname" == ${prefix}-${user}-${host}* ]] ; then # found lock on the requested port, owned by user on current display portopen='true' if (( $verbosity > 0 )) ; then echo "Using socket lock: $lockname" ; fi count=$port break elif [[ "$lockname" == ${prefix}-* ]] ; then echo "Port $port is in use by some other user or a different host" 1>&2 if (( $verbosity > 0 )) ; then echo "Port lock: $lockname" ; fi exit 1 fi done # specified port not in use count=$port # IF EXISTING WINDOW IS REQUESTED, LOOK FOR LISTENING PORT elif [[ "$newwindow" == 'false' && ${#locklist} != 0 ]] ; then # look for a lock on the current display for this user for lockname in ${locklist[*]} ; do if [[ "$lockname" == [0-9][0-9]-${user}-${host}-${display} ]] ; then portopen='true' if (( $verbosity > 0 )) ; then echo "Found socket lock: $lockname" ; fi # if a matching user/display is found, use this one count="${lockname%-*-*-*}" #count=`echo $lockname | sed -e 's/^\([0-9][0-9]\).*/\1/' -e 's/^0//'` # csh? break fi done fi # IF A NEW PORT IS TO BE USED declare -i attempts=0 while [[ "$portopen" == 'false' ]] ; do # new window requested or no matching port found # if port is not specified, look for first free port if (( "$port" == 0 )) ; then if (( ${#locklist} == 0 )) ; then # no active locks - use first port count=1 else # active locks - check each port number so see if it is in use # this is not synchronised!! count=0 inuse='true' while [[ "$inuse" == 'true' ]] ; do count=count+1 prefix=`printf '%02u' $count` inuse='false' for lockname in ${locklist[*]} ; do if [[ "$lockname" == ${prefix}-* ]] ; then inuse='true' fi done done fi fi # CREATING A NEW PORT LOCK prefix=`printf '%02u' $count` lockname=${prefix}-${user}-${host}-${display} [[ -n "$ij_log" ]] && echo -e "$$\t$(date)\tCreating lock\t$lockname" >> "$ij_log" 2>
```

```
/dev/null (( $verbosity > 0 )) && echo -n "creating lock $lockname ... " echo
$$ > $lockname 2> /dev/null if [[ "$(head -n 1 $lockname 2> /dev/null)" == $$
]] ; then # port lock successful trap '\rm -f ${ij_tmp}/$lockname >/dev/null
; [[ -n "$ij_log" ]] && echo -e "$$\t$(date)\tReleasing lock\t$lockname" >>
"$ij_log" 2> /dev/null' EXIT TERM KILL # Quitting ImageJ sends EXIT, as does
a kill/kill -9 # CTRL+C in terminal sends INT + EXIT # System shutdown sends
TERM (+EXIT??) portopen='true' if (( $verbosity > 0 )) ; then echo 'done' ;
fi lockFileCreated='true' if [[ -z "$macrocmd" ]] ; then echo 'Open other
images in this ImageJ panel as follows:' echo " imagej -p $count <image1>
[<image2> ... <imageN>]" fi [[ -n "$ij_log" ]] && echo -e
"$$\t$(date)\tSocket lock:\t$lockname" >> "$ij_log" 2> /dev/null if ((
$verbosity > 0 )) ; then echo "Socket lock: $lockname" ; fi echo else # port
lock unsuccessful (simultaneous instances?) [[ -n "$ij_log" ]] && echo -e
"$$\t$(date)\tAttempted lock failed\t$lockname" >> "$ij_log" 2> /dev/null if
(( $attempts > $max_attempts )) ; then echo "Failed to secure a port lock for
ImageJ after $max_attempts" >&2 [[ -n "$ij_log" ]] && echo -e
"$$\t$(date)\tGiving up\t$lockname" >> "$ij_log" 2> /dev/null fi # REREAD
LOCK LIST declare -a locklist=(`ls | grep '[0-9][0-9]-.*'`) fi
attempts=$attempts+1 done if (( $verbosity > 0 )) ; then echo ${JAVA}
${java_arch} -mx${mem}m ${modules} ij.ImageJ -ijpath ${ij_path} -port${count}
${images} ${macrocmd} ${macroargs} fi cd "$dir" eval "${JAVA} ${java_arch}
-mx${mem}m ${modules} ij.ImageJ -ijpath ${ij_path} -port${count} ${images}
${macrocmd} ${macroargs} " exit 0  / 1024); if (fmem &lt; maxMem) {print
fmem} else {print maxMem}}&#039;`
free_mem=`free | awk -v maxMem=$max_32bit &#039;NR == 3 {fmem=int( / 1024);
if (fmem &lt; maxMem) {print fmem} else {print maxMem}}&#039;`
fi
elif [[ &quot;$OS&quot; == Darwin ]] ; then
java_arch=&#039;-d64&#039;
JAVA_HOME=&quot;${java_home_Darwin:-$java_home}&quot;
max_mem=`vm_stat | awk &#039;NR == 2 {free=} NR == 3 {active=} NR == 4
{inactive=} END{print int((free+active+inactive)/256)}&#039;`
free_mem=`vm_stat | awk &#039;BEGIN{maxMem=&#039;$max_64bit&#039;} NR == 2
{fmem=int( / 256)} NR == 4 {imem=int( / 256)} END{amem=fmem+imem ; if (amem
&lt; maxMem) {print amem} else {print maxMem}}&#039;`
fi
mem=${free_mem}/3*2
(( $mem &gt; $default_mem || $mem &lt; $min_mem )) &amp;&amp;
mem=$default_mem

if [[ -f ${JAVA_HOME}/bin/java ]] ; then
JAVA=${JAVA_HOME}/bin/java
else
JAVA=${JAVA_HOME}/java
unset JAVA_HOME # confuses Mac java
fi

# if tools.jar is not in ${ij_path}/jre/lib/ext/ edit the &#039;tools=&#039;
line
# to point to tools.jar. The -compile switch will load tools.jar into the
# classpath and enable plugins to be compiled in imagej
```

```
if [[ -f &quot;${ij_path}/tools.jar&quot; ]] ; then
tools=&quot;${ij_path}/tools.jar&quot;
else
tools=&#039;&#039;
fi

# End Site specific variables
------------------------------------------------------

# other variables
dir=`pwd`
user=`whoami`
host=`hostname`  # `hostname -s` is better but not solaris 8 compat.
if [[ -z &quot;$DISPLAY&quot; ]] ; then
echo &#039;Display variable not set&#039;
echo &#039;If ImageJ fails to load, try &#039;
echo &#039;% setenv DISPLAY yourcomputer:0&#039;
echo &#039;if you use the &quot;csh&quot; or for &quot;bash&quot; try&#039;
echo &#039;% export DISPLAY=yourcomputer:0&#039;
display=&#039;default&#039;
else
display=&quot;${DISPLAY##*/}&quot;
# display variable on Darwin can look like this:
&#039;/tmp/launch-si9S06/:0&#039;
fi

declare -i port=0
declare -i verbosity=0
declare -i max_attempts=10
images=&#039;&#039;
macrocmd=&#039;&#039;
macroargs=&#039;&#039;

function usage {
echo
echo &#039;Image display and analysis program. Opens formats including:&#039;
echo &#039;UNC, Analyze, Dicom, NIFTI, Tiff, Jpeg, Gif, PNG ...&#039;
echo
echo &#039;imagej [options] image [ image2 ... image3 ]&#039;
echo &#039;    -h        print help and more options&#039;
echo &#039;    -o        open images in an open ImageJ panel&#039;
echo &#039;    -p &lt;N&gt;    open images in ImageJ panel number
&lt;N&gt;&#039;
echo &quot;    -x &lt;MB&gt;   set available memory (default=${mem}
max=${max_mem})&quot;
echo
}

function fullusage {
echo
echo &#039;Image display and analysis program. Opens formats including:&#039;
```

```
echo &#039;UNC, Analyze, Dicom, NIFTI, Tiff, Jpeg, Gif, PNG ...&#039;
echo
echo &#039;imagej [options] image [ image2 ... image3 ] -&gt; open
images&#039;
echo
echo &#039;basic options:&#039;
echo &#039;  -h        print help and more options&#039;
echo &#039;  -o        open images in existing ImageJ panel if one
exists&#039;
echo &#039;  -p &lt;N&gt;    open images in existing ImageJ panel number
&lt;N&gt;&#039;
echo &quot;  -x &lt;MB&gt;   set available memory (default=${mem}
max=${max_mem})&quot;
echo
echo &#039;advanced options:&#039;
echo &#039;  -c        enable plugin compilation within imagej&#039;
echo &#039;  -d        use development version&#039;
echo &quot;  -j        use development java version ($java_home_dev)&quot;
echo &#039;  -v        be verbose (vv or vvv increases verbosity)&#039;
echo
echo &#039;options for batch processing:&#039;
echo &quot;  -e &#039;Macro Code&#039;            execute macro code&quot;
echo &quot;  -r &#039;Menu Command&#039;          run menu command&quot;
echo &quot;Quotation marks &#039;&#039; are required around commands
including spaces&quot;
echo &#039;Commands can be sent to open ImageJ panels with the -p option&#039;
echo
echo &#039;options for macros:&#039;
echo &#039;imagej [-i image] [-b|-m] [arg1 ... argN] &#039;
echo &#039;  -b macro    run macro without graphics window&#039;
echo &#039;  -m macro    run macro&#039;
echo &#039;&quot;image&quot; will be opened before macro is run&#039;
echo &#039;all following arguments are passed to macro&#039;
echo
echo &quot;Documentation - $doc_url &quot;
echo &quot;Report problems with this software to $ijadmin&quot;
echo
}

function macroCmdError {
fullusage
echo &#039;Only one command option (-b -e -m OR -r) may be specified&#039;
1&gt;&amp;2
echo &quot;Macro commands can&#039;t be used with the &#039;-o&#039;
option&quot;  1&gt;&amp;2
exit 1
}

# parse input arguments
while getopts b:cde:hi:jm:nop:r:vx: options
do
```

```
case $options in
b)    [[ -n &quot;$macrocmd&quot; ]] &amp;&amp; macroCmdError # exits
macrocmd=&quot;-batch ${OPTARG}&quot;
display=&#039;batch&#039;
newwindow=&#039;false&#039;
;;
c)    modules=&quot;${modules:-}${modules+:}${tools}&quot;
;;
d)    ij_path=&quot;$ij_path_dev&quot;
;;
e)  [[ -n &quot;$macrocmd&quot; ]] &amp;&amp; macroCmdError # exits
macrocmd=&#039;-eval&#039;
macroargs=&quot;&#039;${OPTARG}&#039;&quot;
;;
h)  fullusage
exit 0
;;
i)  images=&quot;${images}&#039;${OPTARG}&#039; &quot;
;;
j)  JAVA_HOME=&quot;$java_home_dev&quot;
;;
m)    [[ -n &quot;$macrocmd&quot; ]] &amp;&amp; macroCmdError # exits
macrocmd=&quot;-macro ${OPTARG}&quot;
;;
n)  newwindow=&#039;true&#039;
;;
o)  [[ -n &quot;$macrocmd&quot; ]] &amp;&amp; macroCmdError # exits
newwindow=&#039;false&#039;
;;
p)    newwindow=&#039;false&#039;
port=&quot;${OPTARG}&quot;
if (( &quot;$port&quot; &lt; 1 || &quot;$port&quot; &gt; 99 )) ; then
echo &quot;${OPTARG} is not a permissible value for port number (-p)&quot;
1&gt;&amp;2
exit 1
fi
;;
r)    [[ -n &quot;$macrocmd&quot; ]] &amp;&amp; macroCmdError # exits
macrocmd=&#039;-run&#039;
macroargs=&quot;&#039;${OPTARG}&#039;&quot;
;;
v)    verbosity=verbosity+1
if (( $verbosity == 2 )) ; then set -x ; fi
if (( $verbosity == 3 )) ; then set -v ; fi
;;
x)    mem=&quot;${OPTARG}&quot;
newwindow=&#039;true&#039;
if (( $mem &lt; $min_mem || $mem &gt; $max_mem )) ; then
echo &quot;${OPTARG} is not a permissible value for memory (-x)&quot;
1&gt;&amp;2
echo &quot;min=${min_mem}, max=${max_mem}&quot; 1&gt;&amp;2
```

```
exit 1
fi
;;
\?) usage
exit 1
;;
esac
done

declare -i i=1
while (( i &lt; $OPTIND )) ; do
shift
i=i+1
done

if [[ &quot;$#&quot; == 0 &amp;&amp; -z &quot;$macrocmd&quot; ]] ; then
usage
fi

# The best way to install .jar libraries required by plugins is to copy them
# to the imagej plugins/jars directory
# Alternatively, either copy them to ${ij_path}/jre/lib/ext/ or add the .jar
# filepath to the modules line below. Paths are separated by a colon
# Classpath must follow command line arguments, as ij_path is dependent on
the -d option

# Resolving ij.jar path.  If ij.jar is a symbolic link to
ij_&lt;version&gt;.jar
# this allows updating ij.jar without crashing running sessions
ij_jar_path=$(ReadLink  ${ij_path}/ij.jar)

for mod_jar in ${ij_path}/lib/*jar  ; do
modules=&quot;${modules:-}${modules+:}$mod_jar&quot;
done
modules=&quot;-cp ${ij_jar_path}:${modules+:}${modules:-}&quot;
#${ij_path}/plugins/jars/dcmie.jar

export LD_LIBRARY_PATH=&quot;${ij_path}/lib/${OS}_$processor&quot;
if (( $verbosity &gt; 0 )) ; then
echo &quot;LD_LIBRARY_PATH=$LD_LIBRARY_PATH&quot;
fi

# -b and -m options only:
# remaining command line arguments are passed as macro arguments
# separated by $separator
if [[ -n &quot;$macrocmd&quot; &amp;&amp; -z &quot;$macroargs&quot; ]] ; then
while (( &quot;$#&quot; &gt; 0 )) ; do
if [[ -z &quot;$macroargs&quot; ]] ; then
macroargs=&quot; class="code"&quot;
else
macroargs=&quot;${macroargs}${separator} class="code"&quot;
```

```
fi
shift
done
macroargs=&quot;&#039;$macroargs&#039;&quot;
# if user hasn&#039;t requested a specific port number
# and it&#039;s not a batch mode macro, set display to &#039;macro&#039;
# to prevent other instances accessing this ImageJ window
if [[ &quot;$display&quot; != &quot;batch&quot; &amp;&amp; &quot;$port&quot;
== 0 ]] ; then
display=&quot;macro&quot;
newwindow=&#039;true&#039; # should be redundant
fi
fi

# PROTECT POSSIBLE SPACES IN IMAGE FILENAMES
if (( &quot;$#&quot; &gt; 0 )) ; then
while (( &quot;$#&quot; &gt; 0 )) ; do
filearg=&quot; class="code"&quot;
if [[ ! -f &quot;$filearg&quot; ]] ; then
echo &quot;Warning: image &#039;$filearg&#039; may not exist&quot; &gt;&amp;2
fi
# full file path required when sending images to running ImageJ panel
if [[ &quot;${newwindow}&quot; == &#039;false&#039; &amp;&amp; -f
&quot;${filearg}&quot; ]]  &amp;&amp; ! expr &quot;${filearg}&quot; :
&#039;/.*&#039; &gt; /dev/null; then
filearg=&quot;$(ReadLink ${filearg})&quot;
fi
images=&quot;${images}&#039;$filearg&#039; &quot;
shift
done
fi

# CREATE IMAGEJ SOCKET-LOCK DIRECTORY IF NON EXISTANT
if [[ ! -d &quot;$ij_tmp&quot; ]] ; then
mkdir &quot;$ij_tmp&quot;
chmod 777 &quot;$ij_tmp&quot;
fi

# CREATE IMAGEJ LOG FILE IF NON EXISTANT
if [[ -n &quot;$ij_log&quot; &amp;&amp; ! -f &quot;$ij_log&quot; ]] ; then
touch &quot;$ij_log&quot;
chmod 666 &quot;$ij_log&quot;
fi

# CREATES A TEMP FILE INDICATING A PORT IS IN USE BY IMAGEJ
cd &quot;$ij_tmp&quot;
declare -i count=1
portopen=&#039;false&#039;
lockFileCreated=&#039;false&#039;
declare -a locklist=(`ls | grep &#039;[0-9][0-9]-.*&#039;`)
```

```
[[ -n &quot;$ij_log&quot; ]] &amp;&amp; echo -e &quot;$$\t$(date)\tNew Window
= $newwindow&quot; &gt;&gt; &quot;$ij_log&quot; 2&gt; /dev/null
[[ -n &quot;$ij_log&quot; ]] &amp;&amp; echo -e &quot;$$\t$(date)\tPort =
$port&quot; &gt;&gt; &quot;$ij_log&quot; 2&gt; /dev/null
[[ -n &quot;$ij_log&quot; ]] &amp;&amp; echo -e &quot;$$\t$(date)\tlocklist:
\n ${locklist[*]}&quot; &gt;&gt; &quot;$ij_log&quot; 2&gt; /dev/null
if (( $verbosity &gt; 0 )) ; then echo -e &quot;locklist: \n
${locklist[*]}&quot; ; fi

# PORT SPECIFIED BY USER
if (( $port &gt; 0 )) ; then
# look for a lock on the port specified
for lockname in ${locklist[*]} ; do
prefix=`printf &#039;%02u&#039; $port`
if [[ &quot;$lockname&quot; == ${prefix}-${user}-${host}* ]] ; then
# found lock on the requested port, owned by user on current display
portopen=&#039;true&#039;
if (( $verbosity &gt; 0 )) ; then echo &quot;Using socket lock:
$lockname&quot; ; fi
count=$port
break
elif [[ &quot;$lockname&quot; == ${prefix}-* ]] ; then
echo &quot;Port $port is in use by some other user or a different host&quot;
1&gt;&amp;2
if (( $verbosity &gt; 0 )) ; then echo &quot;Port lock: $lockname&quot; ; fi
exit 1
fi
done
# specified port not in use
count=$port

# IF EXISTING WINDOW IS REQUESTED, LOOK FOR LISTENING PORT
elif [[ &quot;$newwindow&quot; == &#039;false&#039; &amp;&amp; ${#locklist}
!= 0 ]] ; then
# look for a lock on the current display for this user
for lockname in ${locklist[*]} ; do
if [[ &quot;$lockname&quot; == [0-9][0-9]-${user}-${host}-${display} ]] ;
then
portopen=&#039;true&#039;
if (( $verbosity &gt; 0 )) ; then echo &quot;Found socket lock:
$lockname&quot; ; fi
# if a matching user/display is found, use this one
count=&quot;${lockname%-*-*-*}&quot;
#count=`echo $lockname | sed  -e &#039;s/^\([0-9][0-9]\).*/
class="code"/&#039; -e &#039;s/^0//&#039;` # csh?
break
fi
done
fi

# IF A NEW PORT IS TO BE USED
```

```
declare -i attempts=0
while [[ &quot;$portopen&quot; == &#039;false&#039; ]] ; do
# new window requested or no matching port found
# if port is not specified, look for first free port
if (( &quot;$port&quot; == 0 )) ; then
if (( ${#locklist} == 0 )) ; then
# no active locks - use first port
count=1
else
# active locks - check each port number so see if it is in use
# this is not synchronised!!
count=0
inuse=&#039;true&#039;
while [[ &quot;$inuse&quot; == &#039;true&#039; ]] ; do
count=count+1
prefix=`printf &#039;%02u&#039; $count`
inuse=&#039;false&#039;
for lockname in ${locklist[*]} ; do
if [[ &quot;$lockname&quot; == ${prefix}-* ]] ; then
inuse=&#039;true&#039;
fi
done
done
fi
fi
# CREATING A NEW PORT LOCK
prefix=`printf &#039;%02u&#039; $count`
lockname=${prefix}-${user}-${host}-${display}

[[ -n &quot;$ij_log&quot; ]] &amp;&amp; echo -e &quot;$$\t$(date)\tCreating
lock\t$lockname&quot; &gt;&gt; &quot;$ij_log&quot; 2&gt; /dev/null
(( $verbosity &gt; 0 )) &amp;&amp; echo -n &quot;creating lock $lockname ...
&quot;
echo $$ &gt; $lockname 2&gt; /dev/null
if [[ &quot;$(head -n 1 $lockname 2&gt; /dev/null)&quot; == $$ ]] ; then #
port lock successful
trap &#039;\rm -f ${ij_tmp}/$lockname &gt;/dev/null ; [[ -n
&quot;$ij_log&quot; ]] &amp;&amp; echo -e &quot;$$\t$(date)\tRereleasing
lock\t$lockname&quot; &gt;&gt; &quot;$ij_log&quot; 2&gt; /dev/null&#039; EXIT
TERM KILL
# Quitting ImageJ sends EXIT, as does a kill/kill -9
# CTRL+C in terminal sends INT + EXIT
# System shutdown sends TERM (+EXIT??)

portopen=&#039;true&#039;

if (( $verbosity &gt; 0 )) ; then  echo &#039;done&#039; ; fi

lockFileCreated=&#039;true&#039;
if [[ -z &quot;$macrocmd&quot; ]] ; then
echo &#039;Open other images in this ImageJ panel as follows:&#039;
```

```
echo &quot;  imagej -p $count &lt;image1&gt; [&lt;image2&gt; ...
&lt;imageN&gt;]&quot;
fi
[[ -n &quot;$ij_log&quot; ]] &amp;&amp; echo -e &quot;$$\t$(date)\tSocket
lock:\t$lockname&quot; &gt;&gt; &quot;$ij_log&quot; 2&gt; /dev/null
if (( $verbosity &gt; 0 )) ; then echo &quot;Socket lock: $lockname&quot; ;
fi
echo
else # port lock unsuccessful (simultaneous instances?)
[[ -n &quot;$ij_log&quot; ]] &amp;&amp; echo -e &quot;$$\t$(date)\tAttempted
lock failed\t$lockname&quot; &gt;&gt; &quot;$ij_log&quot; 2&gt; /dev/null
if (( $attempts &gt; $max_attempts )) ; then
echo &quot;Failed to secure a port lock for ImageJ after $max_attempts&quot;
&gt;&amp;2
[[ -n &quot;$ij_log&quot; ]] &amp;&amp; echo -e &quot;$$\t$(date)\tGiving
up\t$lockname&quot; &gt;&gt; &quot;$ij_log&quot; 2&gt; /dev/null
fi
# REREAD LOCK LIST
declare -a locklist=(`ls | grep &#039;[0-9][0-9]-.*&#039;`)
fi
attempts=$attempts+1
done

if (( $verbosity &gt; 0 )) ; then
echo ${JAVA} ${java_arch} -mx${mem}m ${modules} ij.ImageJ -ijpath ${ij_path}
-port${count} ${images} ${macrocmd} ${macroargs}
fi

cd &quot;$dir&quot;
eval &quot;${JAVA} ${java_arch} -mx${mem}m ${modules} ij.ImageJ -ijpath
${ij_path} -port${count} ${images} ${macrocmd} ${macroargs} &quot;
exit 0
```