

```
#!/usr/bin/perl

#####
# run_radmind
#
# This script runs radmind and reboots.
#
# Requires the radmind client tools (and a running server obviously).
# (See http://rsug.itd.umich.edu/software/radmind)
#
# Can also be used with iHook
# (See http://rsug.itd.umich.edu/software/ihook)
#
# Copyright (c) 2002 University of Utah Student Computing Labs.
# All Rights Reserved.
#
# Permission to use, copy, modify, and distribute this software and
# its documentation for any purpose and without fee is hereby granted,
# provided that the above copyright notice appears in all copies and
# that both that copyright notice and this permission notice appear
# in supporting documentation, and that the name of The University
# of Utah not be used in advertising or publicity pertaining to
# distribution of the software without specific, written prior
# permission. This software is supplied as is without expressed or
# implied warranties of any kind.
#
#####

#####
# change these

$rsrserver = "-h 172.20.0.10";
$scksum = ""; # change to "" to disable
$fsdiffpath = "/"; # change to "/" if your setup has that path

$ktcheckoutput = "/var/log/ktcheck_output.log";
$ktcheckoutput_backup_number = 5;
$fsdiffoutput = "/var/log/fsdiff_output.T";
$fsdiffoutput_backup_number = 5;
$lapplyoutput = "/var/log/lapply_output.log";
$lapplyoutput_backup_number = 5;

$max_tries = 1;

# customize here:
# $radmind_error = "/path/triggerFiles/radmind_error";
$radmind_log = "/var/log/run_radmind";
$errorlog = "/var/log/radmind_error.log";

$ktcheck = "/usr/local/bin/ktcheck -c sha1 $rsrserver 2> \"\$errorlog\"";
$fsdiff = "/usr/local/bin/fsdiff -A $scksum $fsdiffpath 2> \"\$errorlog\"";
$lapply = "/usr/local/bin/lapply -F $scksum $rsrserver $fsdiffoutput 2> \"\$errorlog\"";

#####
&main;

#####
```

```
sub main {

#####
# Turn buffering off
#
$|++;
$oldhandle = select( STDERR );
$|++;
select( $oldhandle );

#####
# Write to log
#

system "echo \"-----\" >> $radmind_log";
system "date >> $radmind_log";
system "echo \"run_radmind started\" >> $radmind_log";

#####
# Roll logs
#
&roll_log($ktcheckoutput, $ktcheckoutput_backup_number);
&roll_log($fsdiffoutput, $fsdiffoutput_backup_number);
&roll_log($lapplyoutput, $lapplyoutput_backup_number);

#####
# Delete stuff ignored by radmind, but still need to be cleaned up.
#
system "/usr/bin/chflags -R nouchg /Library/Caches/com.apple.dock* >> $radmind_log";
system "/bin/rm -rf /Library/Caches/com.apple.dock* >> $radmind_log";
system "/usr/bin/chflags -R nouchg /Library/Caches/com.apple.* >> $radmind_log";
system "/bin/rm -rf /Library/Caches/com.apple.* >> $radmind_log";
system "/usr/bin/chflags -R nouchg /Library/Caches/* >> $radmind_log";
system "/bin/rm -rf /Library/Caches/* >> $radmind_log";

# /System/Library/Caches ?

# prep stuff
chdir ("/");

$current_try = 1;
while (1) {

#####
# run ktcheck
#
print "Checking for Updates\n";
$result = execute_command ("$ktcheck", $ktcheckoutput);
if ($result == 0) {
    print "No updates\n";
} elsif ($result == 1) {
    print "Updates found\n";
} else {
    print "ktcheck encountered a fatal error: $result\n";
    system "echo \"ktcheck encountered a fatal error: $result\" >> $radmind_log";
    &radmindFailed;
}
}
}
```

```
#####  
# run fsdiff  
#  
print "Scanning File System\n";  
$result = execute_command ("fsdiff", $fsdiffoutput);  
  
if ($result != 0) {  
    print "fsdiff encountered a fatal error: $result\n";  
    system "echo \"fsdiff encountered a fatal error: $result\" >> $radmind_log";  
    &radmindFailed;  
}  
  
#####  
# run lapply  
#  
print "Updating File System\n";  
$fsdsize = ( stat( $fsdiffoutput ))[ 7 ];  
if ( $fsdsize > 0 ) {  
    $result = execute_command ("lapply", $lapplyoutput);  
    if ($result == 0) {  
        &radmindSuccessful;  
        last;  
    } elsif ($result == 1) {  
        if ($current_try >= $max_tries) {  
            print "lapply failed too many times: $result\n";  
            system "echo \"lapply failed too many times: $result\" >> $radmind_log";  
            &radmindFailed;  
        } else {  
            print "lapply failed, trying again: $result\n";  
            system "echo \"lapply failed, trying again: $result\" >> $radmind_log";  
            next;  
        }  
    } else {  
        print "lapply encountered a fatal error - see log for errors.\n";  
        system "echo \"lapply failed: $result\" >> $radmind_log";  
        &radmindFailed;  
    }  
} else {  
    print "No changes found!\n";  
    system "echo \"No changes found\" >> $radmind_log";  
    &radmindSuccessful;  
    last;  
}  
$current_try++;  
}  
  
#####  
# This subroutine executes the radmind commands and saves the output to a log,  
# prints it to stdout, and saves stderr to a log as well.  
#  
sub execute_command {  
    local ($thecommand, $path_to_log) = @_;  
    open (COMMAND, "$thecommand |");  
    open (LOG, ">>$path_to_log");  
    while (<COMMAND) {  
        print STDERR;
```

```
    print LOG;
}
close (LOG);
close (COMMAND);
return $? >> 8;
}

#####
# If radmind was successful, do these things
#
sub radmindSuccessful {
    print "radmind finished.\n";
    print "Checking in.\n";
    system "curl http://172.20.4.31/cgi-bin/radmindcheckin";
    print "Restarting...\n";
    system "echo \"run_radmind restarting\" >> $radmind_log";
    sleep 2;
    system "/sbin/reboot";
    exit 0;
}

#####
# If radmind died, this is the cleanup subroutine.
#
sub radmindFailed {
    # print error log to stdout
    open (LOGGY, "<$errorlog");
    while (<LOGGY>) {
        print;
    }
    close (LOGGY);

    system "/sbin/reboot";

    exit 1;
}

#####
# This script renames $path_to_log so that the lowest number is always the
# newest.  The oldest that is greater than $number_of_backups is deleted.
#
sub roll_log {
    local ($path_to_log, $number_of_backups) = @_;
    if (-e $path_to_log) {
        if (-e $path_to_log.".bak$number_of_backups") {
            unlink $path_to_log.".bak$number_of_backups";
        }
        for ($i = $number_of_backups ; $i > 1 ; $i-- ) {
            if (-f $path_to_log.".bak".($i-1)) {
                system "/bin/mv -f \"$path_to_log\".\".bak\".($i-1).\" \"\$path_to_log\".\".bak$i\"";
            }
        }
        system "/bin/mv -f \"$path_to_log\" \"\$path_to_log\".\".bak1\"";
    }
}
}
```