Process Monitor HOW−TO for Linux

Table of Contents

Process Monitor HOW−TO for Linux

Al Dev (Alavoor Vasudevan) alavoor[AT]yahoo.com

v11.3, 12 Feb 2002

This document describes how to monitor Linux/Unix processes and to re−start them automatically if they die without any manual intervention. This document also has URLs for "Unix Processes" FAQs.

11. Copyright Notice

1. Linux or Unix Processes

(The latest version of this document is at http://www.milkywaygalaxy.freeservers.com. You may want to check there for changes).

Processes are the "heart" of the Linux/Unix processes. It is very important to monitor the application processes to ensure 100% availability and reliability of the computer system. For example, processes of databases, web−server etc.. need to be up and running 24 hours a day and 365 days a year. Use the tools described in this document to the monitor important application processes.

See also the following related topics on Linux/Unix processes.

- Unix Programming FAQ − Chapter 1 Unix Processes http://www.erlenstar.demon.co.uk/unix/faq_toc.html

- Other FAQs on Unix are at http://www.erlenstar.demon.co.uk/unix/

2. Unix/Linux command − procautostart

Use the program **procautostart** (say "Prok−Auto−Start" or Process AutoStart) to monitor and automatically re−start any Unix/Linux process if they die. This tiny program is very powerful and is comparable to big commercial products which **costs about $80,000US**. Procautostart can be used for controlling following applications:

- For real−time control of process industries like chemical, manufacturing, power generation and others. Use *nano−seconds* in program to get fine control.
- For controlling processes of software applications like Web servers, database servers, mission critical unix processes, etc..
- As an alarm system for any general monitoring software system. The program can fire a pager or call cell phone or flash red lights on the computer screen. For calling a phone line you may need to use a Telephone card on PCI slot of the computer.

The program listing is given in following sections in this document.

procautostart −n < *delay_seconds* > **−c** "< *command_line* >" nohup &

This starts the unix process **procautostart** and also **command_line** process. The **procautostart** process will re−start **command_line** process if it dies. The −*n* option is the time delay in seconds before **procautostart** checks the running process started by **command_line**. It is advisable to start the procautostart as background process with no−hangup using "nohup &". See 'man nohup'.

The procautostart is written in "C" so that it is very fast and efficient, since the program is called every *n* seconds. Amount of resources consumed by procautostart is **very minute** and is negligible since the program size is small and is highly optimized with −o3 compiler option.

For example −

```
        procautostart −n 12 −c "monitor_test −d $HOME  −a dummy_arg " nohup &
```

Here **procautostart** will be checking the process monitor_test **every** 12 seconds.

The program will output log files in 'mon' sub−directory which has datetime stamp of when the processes died and re−started. These files gives info on how often the processes are dying.

You can also use micro−seconds option '−m' or nano−seconds option '−o', edit the source code file **procautostart.cpp** and uncomment appropriate lines.

3. File procautostart.cpp

Download the latest version of the program from http://www.milkywaygalaxy.freeservers.com, go here and click on 'Source code for Process Monitor HOWTO'.

4. File debug.cpp

// From your browser save this file as **text−file** named as 'debug.cpp'.

```
#ifdef DEBUG_PRT

#include "debug.h"
// Variable value[] can be char, string, int, unsigned long, float, etc...

void local_dbg(char name[], char value[], char fname[], int lineno, bool logfile) {
        printf("\nDebug %s Line: %d %s is = %s\n", fname, lineno, name, value ); }

void local_dbg(char name[], int value, char fname[], int lineno, bool logfile) {
        printf("\nDebug %s Line: %d %s is = %d\n", fname, lineno, name, value ); }

void local_dbg(char name[], unsigned int value, char fname[], int lineno, bool logfile) {
        printf("\nDebug %s Line: %d %s is = %d\n", fname, lineno, name, value ); }

void local_dbg(char name[], long value, char fname[], int lineno, bool logfile) {
        printf("\nDebug %s Line: %d %s is = %ld\n", fname, lineno, name, value ); }

void local_dbg(char name[], unsigned long value, char fname[], int lineno, bool logfile) {
        printf("\nDebug %s Line: %d %s is = %ld\n", fname, lineno, name, value ); }

void local_dbg(char name[], short value, char fname[], int lineno, bool logfile) {
        printf("\nDebug %s Line: %d %s is = %d\n", fname, lineno, name, value ); }

void local_dbg(char name[], unsigned short value, char fname[], int lineno, bool logfile) {
        printf("\nDebug %s Line: %d %s is = %d\n", fname, lineno, name, value ); }

void local_dbg(char name[], float value, char fname[], int lineno, bool logfile) {
        printf("\nDebug %s Line: %d %s is = %f\n", fname, lineno, name, value ); }
```

```
void local_dbg(char name[], double value, char fname[], int lineno, bool logfile) {
        printf("\nDebug %s Line: %d %s is = %f\n", fname, lineno, name, value ); }

// You add many more here − value can be a class, ENUM, datetime, etc...

#endif // DEBUG_PRT
```

5. File debug.h

// From your browser save this file as **text−file** named as 'debug.h'.

```
#ifdef DEBUG_PRT

#include <stdio.h>
//#include <strings.h>
//#include <assert.h>  // assert() macro which is also used for debugging

// Debugging code
// Use debug2_ to output result to a log file
#define debug_(NM, VL) (void) ( local_dbg(NM, VL, __FILE__, __LINE__) )
#define debug2_(NM, VL, LOG_FILE) (void) ( local_dbg(NM, VL, __FILE__, __LINE__, LOG_FILE) )
void local_dbg(char name[], char value[], char fname[], int lineno, bool logfile= false);
void local_dbg(char name[], int value, char fname[], int lineno, bool logfile= false);
void local_dbg(char name[], unsigned long value, char fname[], int lineno, bool logfile= false);
void local_dbg(char name[], float value, char fname[], int lineno, bool logfile= false);

#else

#define debug_(NM, VL) ((void) 0)
#define debug2_(NM, VL, LOG_FILE) ((void) 0)

#endif // DEBUG_PRT
```

6. Makefile

# From your browser save this file as **text−file** named as 'Makefile'.

```
#//*****************************************************************
#// Copyright policy is GNU/GPL and it is requested that
#// you include author's name and email on all copies
#// Author : Al Dev Email: alavoor[AT]yahoo.com
#//*****************************************************************

.SUFFIXES: .pc .cpp .c .o

HOSTFLAG=−DLinux
#HOSTFLAG=−DSunOS

CC=gcc
```

```
CXX=g++

MAKEMAKE=mm
#LIBRARY=libString.a
DEST=/home/myname/lib

# Note: You should set only ONE value of MYCFLAGS below, that is only
# one line is uncommented and others are commented.
# Use options −Wall (all warning msgs) −O3 (optimization)
MYCFLAGS=−DDEBUG_PRT −g3 −Wall
#MYCFLAGS=−O3 −Wall

#PURIFY=purify −best−effort

SRCS=procautostart.cpp debug.cpp
#HDR=my_malloc.h  String.h StringTokenizer.h File.h debug.h string_multi.h
#LIBOBJS=my_malloc.o String.o StringTokenizer.o File.o debug.o
OBJS=procautostart.o  debug.o
EXE=procautostart

# For generating makefile dependencies..
SHELL=/bin/sh

CPPFLAGS=$(MYCFLAGS) $(OS_DEFINES)
CFLAGS=$(MYCFLAGS) $(OS_DEFINES)

#
# If the libString.a is in the current
# directory then use −L. (dash L dot)
MYLIBDIR=−L$(MY_DIR)/libmy −L.

ALLLDFLAGS= $(LDFLAGS)  $(MYLIBDIR)

COMMONLIBS=−lstdc++ −lm
MYLIBS=−lString
LIBS=$(COMMONLIBS)  $(MYLIBS)

all: $(LIBRARY) $(EXE)

$(MAKEMAKE):
        @rm −f $(MAKEMAKE)
        $(PURIFY) $(CXX) −M  $(INCLUDE) $(CPPFLAGS) *.cpp > $(MAKEMAKE)

$(EXE): $(OBJS) $(LIBRARY)
        @echo "Creating a executable "
        $(PURIFY) $(CC) −o $(EXE) $(OBJS) $(ALLLDFLAGS) $(LIBS)

#$(LIBRARY): $(LIBOBJS)
#       @echo "\n***********************************************"
#       @echo "   Loading $(LIBRARY) ... to $(DEST)"
#       @echo "***********************************************"
#       @ar cru $(LIBRARY) $(LIBOBJS)
#       @echo "\n "

.cpp.o: $(SRCS) $(HDR)
#       @echo "Creating a object files from " $*.cpp " files "
        $(PURIFY) $(CXX) −c  $(INCLUDE) $(HOSTFLAG) $(CPPFLAGS) $*.cpp

.c.o: $(SRCS) $(HDR)
#       @echo "Creating a object files from " $*.c " files "
        $(PURIFY) $(CC) −c $(INCLUDE) $(HOSTFLAG) $(CFLAGS) $*.c
```

```
clean:
        rm −f *.o *.log *~ *.log.old *.pid core err a.out lib*.a afiedt.buf  *.class tags
        rm −f $(EXE)
        rm −f $(MAKEMAKE)
        ln −s ../cpphowto/libString.a .

#%.d: %.c
#       @echo "Generating the dependency file *.d from *.c"
#       $(SHELL) −ec '$(CC) −M $(CPPFLAGS) $< | sed '\''s/$*.o/& $@/g'\'' > $@'
#%.d: %.cpp
#       @echo "Generating the dependency file *.d from *.cpp"
#       $(SHELL) −ec '$(CC) −M $(CPPFLAGS) $< | sed '\''s/$*.o/& $@/g'\'' > $@'

# Must include all the c flags for −M option
#$(MAKEMAKE):
#       @echo "Generating the dependency file *.d from *.cpp"
#       $(CXX) −M  $(INCLUDE) $(CPPFLAGS) *.cpp > $(MAKEMAKE)

include $(MAKEMAKE)
#include $(SRCS:.cpp=.d)
#include $(SRCS:.c=.d)
```

7. Testing the program − monitor_test

From your browser save this file as **text−file** named as 'monitor_test'.

Use this program for testing the 'procautostart' program. For example −

```
        procautostart −n 12 −c "monitor_test −d $HOME  −a dummy_arg " nohup &
```

Here **procautostart** will be checking the process monitor_test **every** 12 seconds.

```
#!/bin/ksh

# Program to test the procautostart

echo "Started the monitor_test ...."
date > monitor_test.log
while :
do
        date >> monitor_test.log
        sleep 2
done
```

Then do a tail command to monitor the output. And simulate the failures of monitor_test programs.

```
        bash$ tail −f monitor_test.log
        bash$ ps −ef | grep monitor_test
See the PID of monitor_test and kill it..
```

```
        bash$ kill −9 < PID of monitor_test >
```

Once you kill the process, you will notice that it immediately comes alive due to procautostart !

8. Other Monitoring Tools

8.1 Unix init command

The **init** command is a cool tool to do simple process monitoring. Add :respawn: entry to your /etc/inittab, if you need procees to be respawned. See the online manual page by typing 'man init' at bash prompt.

8.2 OpenSource Monitoring Tools

On linux systems you can find the following packages. If it is not in the main cdrom than you must check in the contrib cdrom :

- On contrib cdrom **daemontools*.rpm**
- 'top' command **procps*.rpm**
- 'top' command graphic mode **procps−X11*.rpm**
- 'ktop' graphic mode **ktop*.rpm**
- 'gtop' graphic mode **gtop*.rpm**
- 'WMMon' CPU load **wmmon*.rpm**
- 'wmsysmon' monitor **wmsysmon*.rpm**
- 'procmeter' System activity meter **procmeter*.rpm**

To use top commands type at unix prompt −

```

        $ top
        $ ktop
        $ gtop
```

8.3 Monitoring Tool − "daemontools"

Visit the web site of daemontools at http://www.pobox.com/~djb/daemontools.html

To install the daemontools RPM, do −

```
        # rpm −i /mnt/cdrom/daemontools*.html
        # man supervise
```

supervise monitors a service. It starts the service and restarts the service if it dies. The companion svc program stops, pauses, or restarts the service on sysadmin request. The svstat program prints a one−line status report. See man page by 'man supervise'

svc − control a supervised service.

svc changes the status of a supervise−monitored service. dir is the same directory used for supervise. You can list several dirs. svc will change the status of each service in turn.

svstat − print the status of a supervised service.

svstat prints the status of a supervise−monitored service. dir is the same directory used for supervise. You can list several dirs. svstat will print the status of each service in turn.

cyclog writes a log to disk. It automatically synchronizes the log every 100KB (by default) to guarantee data integrity after a crash. It automatically rotates the log to keep it below 1MB (by default). If the disk fills up, cyclog pauses and then tries again, without losing any data. See man page by 'man cyclog'

accustamp puts a precise timestamp on each line of input. The timestamp is a numeric TAI timestamp with microsecond precision. The companion tailocal program converts TAI timestamps to local time. See 'man accustamp'

usually watches a log for lines that do not match specified patterns, copying those lines to stderr. The companion errorsto program redirects stderr to a file. See 'man usually'

setuser runs a program under a user's uid and gid. Unlike su, setuser does not gain privileges; it does not check passwords, and it cannot be run except by root. See 'man setuser'

8.4 Commercial Monitoring Tools

There are commercial monitoring tools available. Check out −

- BMC Patrol for Unix/Databases http://www.bmc.com
- TIBCO corp's Hawk for Unix monitoring http://www.tibco.com
- LandMark corporation
- Platinum corporation
- Treeps − X/Motif Unix Process Visualizer http://www.treeps.org

9. Related URLs

Linux goodies main site is at http://www.milkywaygalaxy.freeservers.com Mirror sites are at − angelfire, geocities, virtualave, Fortunecity, Freewebsites, Tripod, 101xs, 50megs,

10. Other Formats of this Document

This document is published in 14 different formats namely − DVI, Postscript, Latex, Adobe Acrobat PDF, LyX, GNU−info, HTML, RTF(Rich Text Format), Plain−text, Unix man pages, single HTML file, SGML (Linuxdoc format), SGML (Docbook format), MS WinHelp format.

This howto document is located at −

- http://www.linuxdoc.org and click on HOWTOs and search for howto document name using CTRL+f or ALT+f within the web−browser.

You can also find this document at the following mirrors sites −

- http://www.caldera.com/LDP/HOWTO
- http://www.linux.ucla.edu/LDP
- http://www.cc.gatech.edu/linux/LDP
- http://www.redhat.com/mirrors/LDP
- Other mirror sites near you (network−address−wise) can be found at http://www.linuxdoc.org/mirrors.html select a site and go to directory /LDP/HOWTO/xxxxx−HOWTO.html

- You can get this HOWTO document as a single file tar ball in HTML, DVI, Postscript or SGML formats from − ftp://www.linuxdoc.org/pub/Linux/docs/HOWTO/other−formats/ and http://www.linuxdoc.org/docs.html#howto

- Plain text format is in: ftp://www.linuxdoc.org/pub/Linux/docs/HOWTO and http://www.linuxdoc.org/docs.html#howto

- Single HTML file format is in: http://www.linuxdoc.org/docs.html#howto

  Single HTML file can be created with command (see man sgml2html) − sgml2html −split 0 xxxxhowto.sgml

- Translations to other languages like French, German, Spanish, Chinese, Japanese are in ftp://www.linuxdoc.org/pub/Linux/docs/HOWTO and http://www.linuxdoc.org/docs.html#howto Any help from you to translate to other languages is welcome.

The document is written using a tool called "SGML−Tools" which can be got from − http://www.sgmltools.org Compiling the source you will get the following commands like

- sgml2html xxxxhowto.sgml (to generate html file)
- sgml2html −split 0 xxxxhowto.sgml (to generate a single page html file)
- sgml2rtf xxxxhowto.sgml (to generate RTF file)
- sgml2latex xxxxhowto.sgml (to generate latex file)

10.1 Acrobat PDF format

PDF file can be generated from postscript file using either acrobat **distill** or **Ghostscript**. And postscript file is generated from DVI which in turn is generated from LaTex file. You can download distill software from http://www.adobe.com. Given below is a sample session:

```
bash$ man sgml2latex
bash$ sgml2latex filename.sgml
bash$ man dvips
bash$ dvips −o filename.ps filename.dvi
bash$ distill filename.ps
bash$ man ghostscript
bash$ man ps2pdf
bash$ ps2pdf input.ps output.pdf
bash$ acroread output.pdf &
```

Or you can use Ghostscript command **ps2pdf**. ps2pdf is a work−alike for nearly all the functionality of Adobe's Acrobat Distiller product: it converts PostScript files to Portable Document Format (PDF) files. **ps2pdf** is implemented as a very small command script (batch file) that invokes Ghostscript, selecting a special "output device" called **pdfwrite**. In order to use ps2pdf, the pdfwrite device must be included in the makefile when Ghostscript was compiled; see the documentation on building Ghostscript for details.

10.2 Convert Linuxdoc to Docbook format

This document is written in linuxdoc SGML format. The Docbook SGML format supercedes the linuxdoc format and has lot more features than linuxdoc. The linuxdoc is very simple and is easy to use. To convert linuxdoc SGML file to Docbook SGML use the program **ld2db.sh** and some perl scripts. The ld2db output is not 100% clean and you need to use the **clean_ld2db.pl** perl script. You may need to manually correct few lines in the document.

- Download ld2db program from http://www.dcs.gla.ac.uk/~rrt/docbook.html or from Milkyway Galaxy site
- Download the cleanup_ld2db.pl perl script from from Milkyway Galaxy site

The ld2db.sh is not 100% clean, you will get lots of errors when you run

```
        bash$ ld2db.sh file−linuxdoc.sgml db.sgml
        bash$ cleanup.pl db.sgml > db_clean.sgml
        bash$ gvim db_clean.sgml
        bash$ docbook2html db.sgml
```

And you may have to manually edit some of the minor errors after running the perl script. For e.g. you may need to put closing tag < /Para> for each < Listitem>

10.3 Convert to MS WinHelp format

You can convert the SGML howto document to Microsoft Windows Help file, first convert the sgml to html using:

```
        bash$ sgml2html xxxxhowto.sgml     (to generate html file)
        bash$ sgml2html −split 0   xxxxhowto.sgml (to generate a single page html file)
```

Then use the tool HtmlToHlp. You can also use sgml2rtf and then use the RTF files for generating winhelp files.

10.4 Reading various formats

In order to view the document in dvi format, use the xdvi program. The xdvi program is located in tetex−xdvi*.rpm package in Redhat Linux which can be located through ControlPanel | Applications | Publishing | TeX menu buttons. To read dvi document give the command −

```
                xdvi −geometry 80x90 howto.dvi
                man xdvi
```

And resize the window with mouse. To navigate use Arrow keys, Page Up, Page Down keys, also you can use 'f', 'd', 'u', 'c', 'l', 'r', 'p', 'n' letter keys to move up, down, center, next page, previous page etc. To turn off expert menu press 'x'.

You can read postscript file using the program 'gv' (ghostview) or 'ghostscript'. The ghostscript program is in ghostscript*.rpm package and gv program is in gv*.rpm package in Redhat Linux which can be located through ControlPanel | Applications | Graphics menu buttons. The gv program is much more user friendly than ghostscript. Also ghostscript and gv are available on other platforms like OS/2, Windows 95 and NT, you view this document even on those platforms.

- Get ghostscript for Windows 95, OS/2, and for all OSes from http://www.cs.wisc.edu/~ghost

To read postscript document give the command −

```
                        gv howto.ps
                        ghostscript howto.ps
```

You can read HTML format document using Netscape Navigator, Microsoft Internet explorer, Redhat Baron Web browser or any of the 10 other web browsers.

You can read the latex, LyX output using LyX a X−Windows front end to latex.

11. Copyright Notice

Copyright policy is GNU/GPL as per LDP (Linux Documentation project). LDP is a GNU/GPL project. Additional restrictions are − you must retain the author's name, email address and this copyright notice on all the copies. If you make any changes or additions to this document than you should intimate all the authors of this document.