LXPROC – A tool for examining Linux Processes

This tool is based on the XASTOR utility written by Mike Tanzer and is freely available from the IBM VM website. XASTOR is a multi-functioned, programmable, full-screen storage examiner, which can be used to display system storage, CP virtual storage, and storage belonging to other virtual machines. It includes facilities for labeling addresses, chasing chains, locating symbols and control blocks, and address translation. It can be used to examine physical tape and disk records as well as CMS blocks and CP pages on DASD.

XASTOR provides a SUBCOM environment that enables Rexx macros to be written to map storage environments, chase complex chains, and perform other user-designed functions. The SUBCOM environment can be used without a display, enabling Rexx execs to examine storage, read tape and disk records, etc.

LXPROC is a macro that I've written that exploits the XASTOR utility. It allows a user to enter the name of the Linux guest and the address of the task structure of a specific process and to then be able to scroll through the virtual storage of that process.

Using LXPROC

To use this utility, first determine the address of the task structure of the process you are interested in or you can get the details of the "current" process.

For example, if I were interested in a Rexx program running on the user DALI159:

1. Determine the process ID:

```
[usanefe@dali159 - usanefe] ps
  PID TTY          TIME CMD
 8372 pts/1    00:00:01 bash
 8446 pts/1    00:00:01 rexx
 8447 pts/1    00:00:02 ps
```

2. Determine the task structure pointer by examining the contents of `/proc/<pid>/status`:

```
[usanefe@dali159 - usanefe] cat /proc/8446/status
Name:   rexx
State:  R (running)
Pid:    8446
PPid:   8372
TracerPid:      0
Uid:    500     500     500     500
Gid:    101     101     101     101
FDSize: 256
Groups: 101
VmSize:     2112 kB
VmLck:         0 kB
VmRSS:       792 kB
VmData:      380 kB
VmStk:        12 kB
VmExe:       248 kB
VmLib:      1428 kB
SigPnd: 0000000000000000
SigBlk: 0000000000000000
SigIgn: 8000000000000000
SigCgt: 0000000000004003
CapInh: 0000000000000000
CapPrm: 0000000000000000
CapEff: 0000000000000000

User PSW:    070dc000 8040e484
```

```
task: 1476a000 ksp: 1476be80 pt_regs: 1476bf68
User GPRS:
00000019  0046e798  0046e798  0046e7cc
00000000  00000000  00000004  00000000
00425d64  004086c0  00000000  7ffff2b0
0045e528  8040db90  8040dd8a  7ffff228
User ACRS:
00000000  00000000  00000000  00000000
00000001  00000000  00000000  00000000
00000000  00000000  00000000  00000000
00000000  00000000  00000000  00000000
```

At this point I can logon to the Virtual Machine where XASTOR is installed. I invoke the utility by simply entering XASTOR. I am then presented with the following full-screen display:

```
00000000    Label:          User: USANEFE   Next: +00000000 Cursor:
      ==> XASTOR 1.8.0
00000000  030C1000  81F360D6  81F35C18  00000000     *....a3-Oa3*.....*
00000010  00007AE8  00000600  030C0000  00F34C98     *..:Y.........3<q*
00000020  000C1000  80F12368  030C2000  81F37146     *....Ø1.Ç....a3Éã*
00000030  000C2000  80F12128  FFFFFFFF  FFFFFFFF     *....Ø1..........*
00000040  00000000  00000000  00000000  00000000     *................*
00000050  00000000  00000000  00080000  80F22D80     *............Ø2.Ø*
00000060  000C0000  80F11206  00080000  81375548     *....Ø1......a.íç*
00000070  00080000  80F3E3B8  00080000  80F10908     *....Ø3T½....Ø1..*
00000080  00000000  00004001  000200C9  00040005     *...... ....I...;*
00000090  00000000  00000000  00000000  00000000     *................*
000000A0  00000000  00000000  00000000  00000000     *................*
000000B0  00000000  00000000  00010001  00000000     *................*
000000C0  00000000  00000000  00000000  00000000     *................*
000000D0  C2000000  00000000  FFFFF350  8A92EFC0     *B.........3&«kÕ{*
000000E0  00000000  00000000  00400F1D  400B0000     *......... .. ...*
000000F0  00000000  00000000  00000000  00000000     *................*
00000100  00000000  00000000  00000000  00000000     *................*
00000110  00000000  00000000  00000000  00000000     *................*
00000120  00000000  00000000  00000000  00000000     *................*
00000130  00000000  00000000  00000000  00000000     *................*
00000140  00000000  00000000  00000000  00000000     *................*
1)Help 2)? 3)Quit 4)L 5)LH 6)LA 7)Ba 8)Fo 9)Chain 10)Add 11)Drop 12)Cur
```

TAB down to the `==>` line and enter: `LXPROC <task structure pointer | *> <Linux Guest Name>`

```
00000000    Label:          User: USANEFE   Next: +00000000 Cursor:
      ==> LXPROC 1476a000 DALI159
00000000  030C1000  81F360D6  81F35C18  00000000     *....a3-Oa3*.....*
00000010  00007AE8  00000600  030C0000  00F34C98     *..:Y.........3<q*
00000020  000C1000  80F12368  030C2000  81F37146     *....Ø1.Ç....a3Éã*
00000030  000C2000  80F12128  FFFFFFFF  FFFFFFFF     *....Ø1..........*
00000040  00000000  00000000  00000000  00000000     *................*
00000050  00000000  00000000  00080000  80F22D80     *............Ø2.Ø*
00000060  000C0000  80F11206  00080000  81375548     *....Ø1......a.íç*
00000070  00080000  80F3E3B8  00080000  80F10908     *....Ø3T½....Ø1..*
00000080  00000000  00004001  000200C9  00040005     *...... ....I...;*
00000090  00000000  00000000  00000000  00000000     *................*
000000A0  00000000  00000000  00000000  00000000     *................*
000000B0  00000000  00000000  00010001  00000000     *................*
000000C0  00000000  00000000  00000000  00000000     *................*
000000D0  C2000000  00000000  FFFFF350  8A92EFC0     *B.........3&«kÕ{*
000000E0  00000000  00000000  00400F1D  400B0000     *......... .. ...*
000000F0  00000000  00000000  00000000  00000000     *................*
00000100  00000000  00000000  00000000  00000000     *................*
00000110  00000000  00000000  00000000  00000000     *................*
00000120  00000000  00000000  00000000  00000000     *................*
00000130  00000000  00000000  00000000  00000000     *................*
00000140  00000000  00000000  00000000  00000000     *................*
1)Help 2)? 3)Quit 4)L 5)LH 6)LA 7)Ba 8)Fo 9)Chain 10)Add 11)Drop 12)Cur
```

After hitting enter you should end up with something like the following. The message line shows the current PSW, the segment table origin (STO or pgd), the task structure pointer and the user context (which contains the registers).

```
7FFFF228    Label:          User: DALI159   Next: +00000000 Cursor:
      ==> PSW: 070DD000 C03EEDE2 STO: 0DFD4000 TSS: 1476A000 CTX: 1476BF68
7FFFF220  00460558  7FFFF8C4  7FFFF2E0  0046A5E0     *.F.X.K...K...F..*
7FFFF230  0043F278  00000000  00000000  8042DC2C     *.C.x.........B.,*
7FFFF240  00460558  7FFFF8C4  00000002  7FFFF340     *.F.X.K.......K.@*
7FFFF250  00000000  0043F378  7FFFF740  8040DB90     *.....C.x.K.@.@..*
7FFFF260  8040DD8A  7FFFF228  7FFFF2E0  00000000     *.@...K.(.K......*
7FFFF270  7FFFF160  00000005  00000018  00000005     *.K.`............*
7FFFF280  7FFFF2E0  00000354  00000002  7FFFF340     *.K.....T.....K.@*
7FFFF290  FFFFFFFF  00000001  00000000  00000000     *KKKK............*
7FFFF2A0  00000000  00000000  00000000  00000000     *................*
7FFFF2B0  7FFFF1C4  7FFFF250  7FFFF1C0  80424758     *.K...K.P.K...BGX*
7FFFF2C0  C000EFA4  0043F378  0046E798  00000000     *.....C.x.F......*
7FFFF2D0  00000000  00000000  0046E798  0045E520     *.........F...E. *
7FFFF2E0  7FFFF7E8  00000000  00000000  00000000     *.K..............*
7FFFF2F0  00000000  00000000  00460558  7FFFF8C4     *.........F.X.K..*
7FFFF300  00000002  7FFFF340  00000000  0043F378     *.....K.@.....C.x*
7FFFF310  7FFFF740  8042F938  8042FDAC  7FFFF2E0     *.K.@.B.8.B...K..*
7FFFF320  00000001  00000000  00000000  00000000     *................*
7FFFF330  00000000  00000000  7FFFF598  00000000     *.........K......*
7FFFF340  2F46532F  66733033  38632F75  73616E65     */FS/fs038c/usane*
7FFFF350  66652F74  6573742E  72657878  00000000     *fe/test.rexx....*
7FFFF360  00000000  00000000  00000000  00000000     *................*
1)Help 2)? 3)Quit 4)L 5)LH 6)LA 7)Ba 8)Fo 9)Chain 10)Add 11)Drop 12)Cur
```

At this point you are looking at the current stack area (`0x7ffff228`). You can now chain through the stack area by putting the cursor over the backchain field within the current stack from (displacement 0) and pressing F4. Alternatively, I can TAB to the `Next:` field and enter '`@00000000`' and then press F9. This will tell XASTOR to chain through the stack frames using displacement 0 within the frame as the chaining field.

I can also enter any virtual address I'm interested in at the top left field of the screen. Note that the tool automatically displays ASCII characters in the right hand area.

Each time LXPROC is invoked it will extract the PID from the `task_structure` and create a file "<*Linux Guest Name*> <*pid*> A" that contains a dump of the registers and the memory map for the process (see following page).

If you attempt to display a page that is not in the address space of the process you will see something like this:

```
333FF7E8    Label:          User: DALI159   Next: @00000000 Cursor:
      ==> Invalid second-level segment referenced.
```

To display another process's storage, enter `LXPROC <task structure | *>`. The current user will be assumed and if you specify '*****' the currently active process will be examined.

To exit XASTOR press F3 twice.

Sample Process Listing

```
                    Process Context Data

PSW: 070DC000 C0213632
R00: 00000000   R01: 40213630   R02: 00000006   R03: 7FFFF048
R04: 00000000   R05: 00000000   R06: 00000000   R07: 409054E0
R08: 00000001   R09: 00000005   R10: 7FFFF0E0   R11: 7FFFF048
R12: C00F94EC   R13: C00F58CC   R14: C00F5984   R15: 7FFFEFE8

AR00: 40034A80   AR01: 00000000   AR02: 00000000   AR03: 00000000
AR04: 00000001   AR05: 00000000   AR06: 00000000   AR07: 00000000
AR08: 00000000   AR09: 00000000   AR10: 00000000   AR11: 00000000
AR12: 00000000   AR13: 00000000   AR14: 00000000   AR15: 00000000

                        Memory Map

00400000 - 0040D000 : r-xp argevsrv
0040D000 - 0040F000 : rw-p argevsrv
0040F000 - 004CB000 : rwxp
40000000 - 40018000 : r-xp ld-2.2.2.so
40018000 - 40019000 : rw-p ld-2.2.2.so
40019000 - 4001A000 : rwxp
4001A000 - 4001B000 : r--p LC_IDENTIFICATION
4001B000 - 4001C000 : r--p LC_MEASUREMENT
4001C000 - 4001D000 : r--p LC_TELEPHONE
4001D000 - 4001E000 : r--p LC_ADDRESS
4001E000 - 4001F000 : r--p LC_NAME
4001F000 - 40020000 : r--p LC_PAPER
40020000 - 40021000 : r--p SYS_LC_MESSAGES
40021000 - 40022000 : r--p LC_MONETARY
40022000 - 40030000 : r-xp libpthread.so.0
40030000 - 40038000 : rw-p libpthread.so.0
40038000 - 40047000 : r-xp libsagsmp2.so
40047000 - 40049000 : rw-p libsagsmp2.so
40049000 - 40074000 : r-xp libsagovo.so
40074000 - 40075000 : rw-p libsagovo.so
40075000 - 400A6000 : r-xp libsagecs.so
400A6000 - 400AA000 : rw-p libsagecs.so
400AA000 - 400AD000 : rw-p
400AD000 - 400C4000 : r-xp libsagrgs.so
400C4000 - 400C6000 : rw-p libsagrgs.so
400C6000 - 400CC000 : r-xp libcrypt.so.1
400CC000 - 400CD000 : rw-p libcrypt.so.1
400CD000 - 400F4000 : rw-p
400F4000 - 400F9000 : r-xp libargtcp.so
400F9000 - 400FA000 : rw-p libargtcp.so
400FA000 - 400FC000 : r-xp libargtrace.so
400FC000 - 400FD000 : rw-p libargtrace.so
400FD000 - 4010A000 : r-xp libargutl.so
4010A000 - 4010C000 : rw-p libargutl.so
4010C000 - 4010F000 : r-xp libargdev.so
4010F000 - 40111000 : rw-p libargdev.so
40111000 - 4011B000 : rw-p
4011B000 - 4013A000 : r-xp libargevt.so
4013A000 - 4013C000 : rw-p libargevt.so
4013C000 - 4013D000 : rw-p
4013D000 - 4025E000 : r-xp libc.so.6
4025E000 - 40264000 : rw-p libc.so.6
40264000 - 40268000 : rw-p
40268000 - 4026B000 : r-xp libdl.so.2
4026B000 - 4026C000 : rw-p libdl.so.2
4026C000 - 4026D000 : ---p
4026D000 - 4027D000 : rwxp
4027D000 - 4027E000 : ---p
4027E000 - 4028E000 : rwxp
4028E000 - 4028F000 : r--p LC_TIME
4028F000 - 40290000 : r--p LC_NUMERIC
40290000 - 402AB000 : r--p LC_CTYPE
402AB000 - 402AD000 : r-xp ISO8859-1.so
402AD000 - 402AE000 : rw-p ISO8859-1.so
402AE000 - 402D3000 : rw-p
402D3000 - 402E1000 : r-xp libargen.so
402E1000 - 402E3000 : rw-p libargen.so
```

```
402E3000 - 402E4000 : rw-p
402EC000 - 402F7000 : r-xp libnss_files.so.2
402F7000 - 402F8000 : rw-p libnss_files.so.2
402F8000 - 402F9000 : ---p
402F9000 - 40319000 : rwxp
4035B000 - 4035C000 : ---p
4035C000 - 4037C000 : rwxp
404F8000 - 404F9000 : ---p
404F9000 - 40519000 : rwxp
40519000 - 40521000 : r-xp libinoplug.so
40521000 - 40522000 : rw-p libinoplug.so
40522000 - 40523000 : rw-p
4052B000 - 40572000 : r-xp libncurses.so.5.2
40572000 - 4057D000 : rw-p libncurses.so.5.2
4057D000 - 40580000 : rw-p
40580000 - 40587000 : r-xp librt.so.1
40587000 - 40588000 : rw-p librt.so.1
40588000 - 40592000 : rw-p
40592000 - 40593000 : ---p
40593000 - 405B3000 : rwxp
405B3000 - 40600000 : r-xp inomsg.so
40600000 - 4060A000 : rw-p inomsg.so
4060A000 - 4061E000 : r-xp libnsl.so.1
4061E000 - 40620000 : rw-p libnsl.so.1
40620000 - 40622000 : rw-p
40622000 - 40648000 : r-xp libm.so.6
40648000 - 40649000 : rw-p libm.so.6
40649000 - 4064A000 : ---p
4064A000 - 40849000 : rwxp
40900000 - 40908000 : rw-p
40908000 - 40A00000 : ---p
7FFF9000 - 80000000 : rwxp
```