User-Mode-Linux

Mehrere virtuelle Linuxe parallel auf einem Hostlinux

Vortrag im Rahmen der Unix-AG Saarbrücken, 2002-06-12

Was ist User-Mode-Linux?

- UML ist ein System-Call-Emulator / -Wrapper
  im Einzelnen:
  - Prozessmanagement (Mapping der Prozesse ...)
  - Memory (malloc …, Beziehung mit fork …)
  - Block-Devices (Harddisk-Partitions, Loop-Devices, root-fs, swap)
  - Character-Devices (Terminals, Modems, tty-pty-Pairs, Soundkarten)
  - Konsolen (Spezialfall von Character-Devices)
  - Networking
    (protokollunabhängig, z.B. TCP/IP (V4, V6, IPSEC), IPX/SPX, …)

- stellt Prozessen „Innen“ ein binär-kompatibles Runtime-Environment zur Verfügung, d.h. Anwendung laufen ohne Neucompilierung

- UML-Kernel ist ein statisch gelinktes Binary, das unter einem „normalen“ Benutzer läuft und nach „Innen“ den syscall-verarbeitenden Kernel darstellt.

Risiken und Nebenwirkungen *g* / aktuelle Restriktionen:

- Plattform: i386 ist ok, smp-Emulation noch buggy
- sparc-Port in progress, win-Port in progress

- „innen“ nice’n ist buggy (nice to have)

- */proc/loadavg* „innen“ ist relativ sinnfrei bei hoher IO-Beanspruchung (da im realen „äusseren“ Kernel verbratene CPU-Ticks

- bestimmte Releases sind in Produktiv-Environments eindeutig zu bevorzugen

- kleinere Bugs in switched-Networking bzgl. Empfang eigener Pakete (Fixes existieren)

- kleinere Bugs in ethertap-/tun-tap-Networking unter hoher Load (Fixes existieren, Workaround: mtu „innen“ auf 1484 setzen)

How To? (1/2)

- Fertige UML-Kernel's und root-fs'es sind als Packages verfügbar
  Nachteil(e): teilweise unerwünschte Optionen aktiviert, erwünschte fehlen

Besser:

- UML-Kernel-Bau:
  - linux-Sourcen, passenden UML-Patch, uml-Utilities downloaden, auspacken
  - UML-Patch anwenden: *patch –p1 < uml-patch*
  - Sonstige Patches anwenden: z.B. IPSEC (evtl. makefile-Anpassungen)
  - Konfigurieren: *make ARCH=um menuconfig*
    UML ist eine eigene Architektur: ARCH=um, SUBARCH=i386
  - Bauen des Kernels: *make ARCH=um linux*
  - Anpassen der uml-Utilities (Feinschmecker-Spezialitäten – use the Source)
  - Bauen der passenden uml-Utilities (*make* genügt)

How To? (2/2)

- root-fs'es bauen teilautomagisch:
  - mkrootfs
  - UML-Builder
  - gBootRoot

- root-fs'es-bauen händisch:
  - *dd if=/dev/zero of=./umlrootfs bs=1024 count= `expr 1024 \* 500`*
  - *echo y | mkfs.ext2 ./umlrootfs*
  - zusätzlich für Plattenplatz-Knauserer: *tune2fs -r 0 ./umlrootfs*
  - loop-mounten des fs zwecks Installation einer Distribution:
    mount –o loop ./umlrootfs /installmountpoint
  - lvs oder Distribution nach Gusto installieren nach */installmountpoint*
  - danach *umount ./installmountpoint* niemals nie nicht vergessen
  - Verschiedene Anpassungen bzgl. */etc/inittab*, */etc/fstab*, evtl. Networking

Prozesse:

- „aussen“:

```
USER  PID  COMMAND
uml4  4609  /bin/sh /umlroot/uml4/user.sh -v uml4 linux devfs=nomount ……
uml4  4633  /umlroot/uml4/linux [(tracing thread)]
uml4  4652  /umlroot/uml4/linux [(idle thread)]
uml4  4674  /umlroot/uml4/linux [(kernel thread)]
uml4  4675  /umlroot/uml4/linux [(kernel thread)]
uml4  4676  /umlroot/uml4/linux [(kernel thread)]
uml4  4677  /umlroot/uml4/linux [(kernel thread)]
uml4  4678  /umlroot/uml4/linux [(kernel thread)]
uml4  4679  /umlroot/uml4/linux [(kernel thread)]
uml4  4680  /umlroot/uml4/linux [(kernel thread)]
uml4  4682  /umlroot/uml4/linux [init]
uml4  6445  /umlroot/uml4/linux [/sbin/syslogd]
……
```

- „innen“:

```
USER  PID  COMMAND
root    1  init
root    2  [keventd]
root    3  [ksoftirqd_CPU0]
root    4  [kswapd]
root    5  [bdflush]
root    6  [kupdated]
root    7  [khttpd manager]
root  353  /sbin/syslogd -a
root  357  /sbin/klogd -c 1
……
```

Speicher / malloc:

- Zuteilung von Speicher für eine UML:
  ./linux ... ... mem=96M ... ...
- Speicher „innen“ wird „aussen“ zu Filebuffern gemapped!
  - Pfad steuerbar über Env.-Variablen *TMP*, *TEMP* und *TMPDIR*
  - diese „äusseren“ Filebuffer für „innere“ Prozesse sind exklusiv (open-unlink-Strategie)
- tuning „aussen“ möglich durch Verwendung von *tmpfs*:
  /etc/fstab:

  tmpfs /mountpointumlmemory tmpfs size=100MB,nr_inodes1k 0 0

  ...
- sinnvolle Tuning-Priorisierung: (Feedback welcome 8-))
  Kommandozeilenzuteilung, „äusserer“ *tmpfs*-Pfad, „innerer“ Swap, „äusserer“ *bdflush*, „äusserer“ Swap

Block-Devices

- Mapping:
  - *./linux ubd0=/umlaussen/umlrootfs* (ein Directory!)
  - *./linux ubd0=/umlaussen/umlrootfs* (ein File)
  - Optionen für initrd und andere Spezialitäten (devpts-fs: *devfs=nomount*)
- „aussen“
  - Directorys
  - Filesysteme in Files ge-*loop*-ed, Optionen: *fakehd*, *fake_ide*
  - schablonierte Filesysteme: cowfs (*./linux … ... ubd0=fscow,fsbase ... …*) (Tool zum Mergen: *uml_moo*)
  - reale Devices *(/dev/cdrom*)
- „innen“
  - ubd0-ubd7 (mit Patches mehr, auch Partitiontables möglich), swap möglich, Spezialitäten möglich (*hostfs*: *mount* eines „äusseren“ Directorys) */etc/fstab*-Zeile: */dev/ubd0 / ext2 defaults 1 1*

Character-Devices /Konsolen:

- Mapping(s):
  ./linux … con0=fd:0,fd:1 … con=pty … ssl=pty … </dev/ptyzc &

- “aussen”:
  - File-Deskriptoren (auch *stdin* und *stdout*): *con0=fd:0,fd:1*
  - *xterm*’s: *=xterm*
  - TCP-Ports: *=port:2342*
  - *tty-pty*-Pairs: *=pty* (Zugriff: *minicom -o -p ttyp2*)
  - das Nichts: *=null*
  - Character-Devices (“reale Konsolen”): *=tty:/dev/tty2*

- “innen”:
  - Konsolen: */etc/inittab*: *1:12345:respawn:/sbin/agetty -w 38400 tty1 vt102*
  - Serial-Lines, Modems:
    /etc/inittab: S0:12345:respawn:/sbin/agetty -L 38400 cua0 vt102

Networking:

- ethertap (*uml_net*)
- TUN/TAP (*uml_net*)
- Multicast
- Switch-Daemon (*uml_switch*)
- Slip (*uml_net*)

Networking – ethertap

- Mapping: *./linux … … eth0=ethertap,tap4,fe:fd:0:0:0:4 … …*

- “aussen”:

insmod ethertap unit=${UMLTAPID} -o ethertap${UMLTAPID}

*ifconfig tap${UMLTAPID} arp mtu 1484 ${UMLTAPIP} *
* netmask 255.255.255.255 up*

route add -host ${UMLUMLETHIP} gw ${UMLTAPIP} dev tap${UMLTAPID}

arp -Ds ${UMLUMLETHIP} eth0 pub

Ein suid-Helper (!) (*uml_net*) stellt die Verbindung zwischen *ethertap* und UML-Prozess-Tree her

- “innen”:

ganz normal, eventuell abnorme Hostmasks/Broadcast-Adressen berücksichtigen

Networking – virtuelles Switching

- Mapping:

./linux … eth1=daemon,fe:fd:0:0:0:11,unix,/tmp/uml-vswitch0.ctl,
/tmp/uml-vswitch0.data

- „aussen“:

uml_switch -unix /tmp/uml-vswitch0.ctl /tmp/uml-vswitch0.data </dev/null &

/bin/chown swgr0.swgrp /tmp/uml-vswitch0.ctl /tmp/uml-vswitch0.data

/bin/chmod 770 /tmp/uml-vswitch0.ctl /tmp/uml-vswitch0.data

- “innen”:

wie im echten Leben

Security-Überlegungen

- UML in root-jail
- Jail-Option von UML
- nicht-modularer UML-Kernel
- Kein Host-FS im UML-Kernel
- Nicht-modularer Wirts-System-Kernel, zumindest keine anderen Syscall-Emulatoren (xenix-, sco-Emulation) im Wirts-System

  in Frage kommende Übeltäter:
  /lib/modules/`uname –r`/kernel/abi/[common|solaris|svr4|uw7]

Literatur/WWW-Links:

- Home-Page User-Mode-Linux: http://user-mode-linux.sourceforge.net
- UML-Patches, fertige UML-Kernels, fertige UML-rootfs-es, Utilities: http://user-mode-linux.sourceforge.net/dl-sf.html
- Linux-Kernel-Sources z.B.: ftp://ftp.kernel.org/pub/linux/kernel/v2.4/
- rootfs-Build-Tools:
  - mkrootfs: http://www.stearns.org/mkrootfs/
  - UML-Builder: http://umlbuilder.sourceforge.net/
  - gBootRoot: http://gbootroot.sourceforge.net/
- Mailinglist User-Mode-Linux User: http://www.geocrawler.com/redir-sf.php3?list=user-mode-linux-user
- Mailinglist User-Mode-Linux Developer: http://lists.sourceforge.net/lists/listinfo/user-mode-linux-devel
- ein User-Mode-Linux-basiertes System: http://matrix.saar.de (18 aktive UML's)