Kernel Debugging User-Space API Library (KDUAL)

John Livingston, TJHSST Computer Systems Research 2005

Background

The Linux kernel is an extremely complex program spanning more than 2 million lines. It must be held to the most stringent standards of performance, as any malfunction, or worse, security flaw, could be potentially fatal for a critical application. However, because of the nature of the kernel and its close interaction with hardware, it's extremely difficult to debug kernel code. The goal of this project is to create a C library that provides the kernel API, but operates in ordinary user space, without actual interaction with the underlying system. Kernel code currently being tested can then be compiled against this library for testing without the risks and confusion of testing it on a live system.

Abstract

The purpose of this project is to create an implementation of much of the kernel API that functions in user space, the normal environment that processes run in. The issue with testing kernel code is that the live kernel runs in kernel space, a separate area that deals with hardware interaction and management of all the other processes. Kernel space debuggers are unreliable and very limited in scope; a kernel failure can hardly dump useful error information because there's no operating system left to write that information to disk.

Kernel development is quite likely the most important active project in the Linux community. Any aids to the development process would be appreciated by the entire kernel development team, allowing them to do their work faster and pass changes along to the end user quicker. This program will make a direct contribution to kernel developers, but an indirect contribution to every future user of Linux.

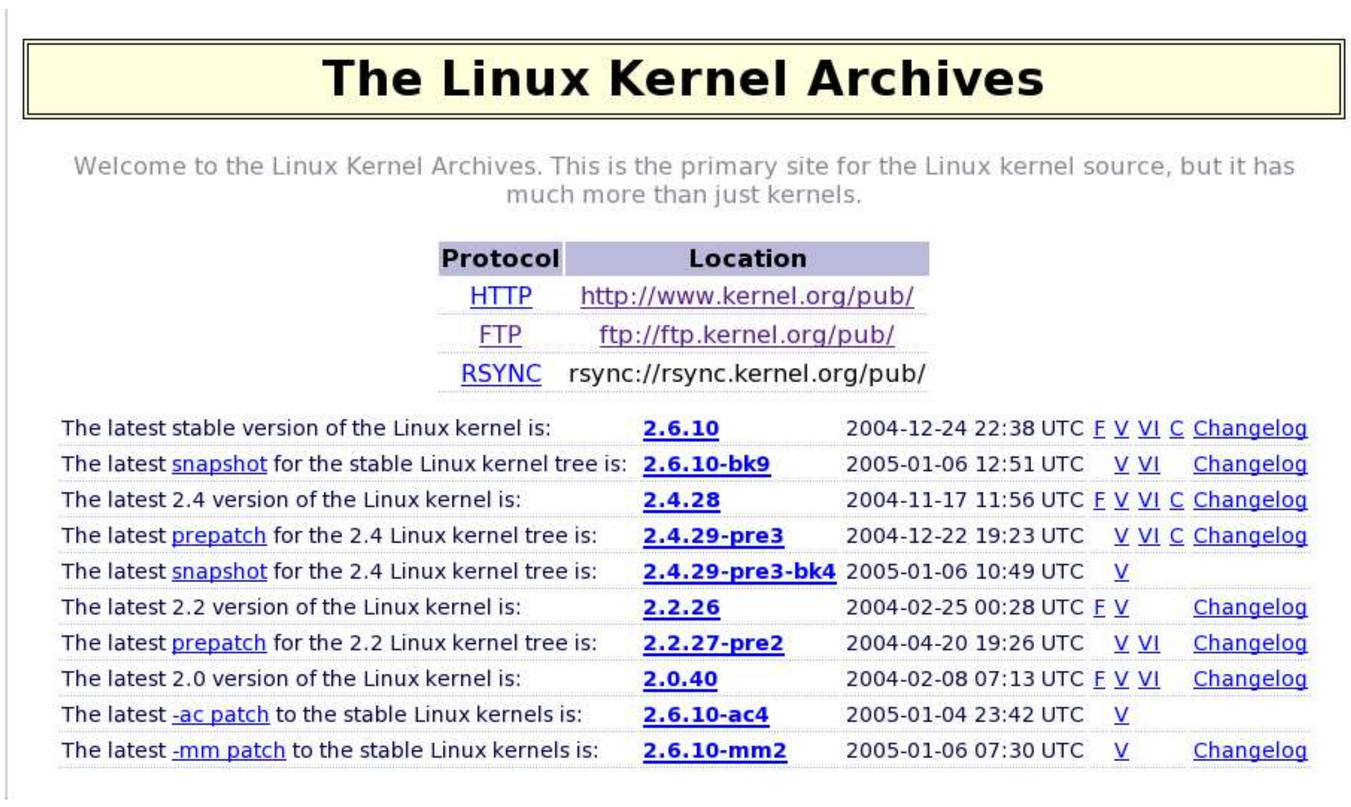
Process

The development of the core library consisted of a great deal of research into the pure software components of the kernel itself. Most development was done using a vanilla 2.6.9 kernel tree, provided by kernel.org, with some additions from 2.6.10. Once all hardware interactions were stripped from the kernel, the necessary sources and headers were recreated within the library. Additional work was done to improve the speed of the library's “hardware” mathematics and data structure manipulation, in order to minimize the inherent time delay caused by simulating on-CPU calcuation with a software program.

Conclusion

The development of this library allows greater ease of debugging for some kernel modules. These modules by necessity must not have any hardware interactions, but beyond that there is no theoretical limit to what the library can be used to test. In its present form the library is only effective on very simplistic modules, but can be expanded to include far more complex interactions.

The natural outgrowth of this project would be its continued development to support different types of kernel modules, possibly by means of a future developer selecting a module they wish to have supported, and add the necessary code to the library to support that module. Using the traditional open source method, perhaps by placing the library on a site such as Sourceforge, support could expand rapidly.



The Linux Kernel Archives

Welcome to the Linux Kernel Archives. This is the primary site for the Linux kernel source, but it has much more than just kernels.

Protocol	Location
HTTP	http://www.kernel.org/pub/
FTP	ftp://ftp.kernel.org/pub/
RSYNC	rsync://rsync.kernel.org/pub/

The latest stable version of the Linux kernel is:	**2.6.10**	2004-12-24 22:38 UTC	F	V	VI	C	Changelog
The latest snapshot for the stable Linux kernel tree is:	**2.6.10-bk9**	2005-01-06 12:51 UTC		V	VI		Changelog
The latest 2.4 version of the Linux kernel is:	**2.4.28**	2004-11-17 11:56 UTC	F	V	VI	C	Changelog
The latest prepatch for the 2.4 Linux kernel tree is:	**2.4.29-pre3**	2004-12-22 19:23 UTC		V	VI	C	Changelog
The latest snapshot for the 2.4 Linux kernel tree is:	**2.4.29-pre3-bk4**	2005-01-06 10:49 UTC		V			
The latest 2.2 version of the Linux kernel is:	**2.2.26**	2004-02-25 00:28 UTC	F	V			Changelog
The latest prepatch for the 2.2 Linux kernel tree is:	**2.2.27-pre2**	2004-04-20 19:26 UTC		V	VI		Changelog
The latest 2.0 version of the Linux kernel is:	**2.0.40**	2004-02-08 07:13 UTC	F	V	VI		Changelog
The latest -ac patch to the stable Linux kernels is:	**2.6.10-ac4**	2005-01-04 23:42 UTC		V			
The latest -mm patch to the stable Linux kernels is:	**2.6.10-mm2**	2005-01-06 07:30 UTC		V			Changelog

http://www.kernel.org/

```
# $(1)  == type
# $$(1) == name
# $$(2) == objects
override define FUNC_BUILD_DEF
$(define__) FUNC_BUILD_$(1)

$(subst $$$$(1),$$(1),$(subst $$,$$$$,  \
    $(call FUNC_GEN_REAL_VAR,$(1),$$(1),DEPFLAGS,)))
$(subst $$$$(1),$$(1),$(subst $$,$$$$,  \
    $(call FUNC_GEN_REAL_VAR,$(1),$$(1),CFLAGS,)))
$(subst $$$$(1),$$(1),$(subst $$,$$$$,  \
    $(call FUNC_GEN_REAL_VAR,$(1),$$(1),CPPFLAGS,)))
$(subst $$$$(1),$$(1),$(subst $$,$$$$,  \
    $(call FUNC_GEN_REAL_VAR,$(1),$$(1),LDFLAGS,)))
$(subst $$$$(1),$$(1),$(subst $$,$$$$,  \
    $(call FUNC_GEN_REAL_VAR,$(1),$$(1),LIBS,)))

lib/$(1)/$$(1)$$$$(CONFIG_LIB_$(1)): configured $$(2)
    install -d $$$$(dir $$$$@)
    $$$$(strip libtool --mode=link $$$$(CC)     \
        $$$$(REAL_$(1)_$$(1)_LDFLAGS)      \
        $$$$(REAL_$(1)_$$(1)_LIBS)      \
        -rpath $$$$(MY_$(1)_$$(1)_DESTDIR)  \
        -o $$$$@ $$(2))

override MASTER_TARGETS += dst/$$$$(MY_$(1)_$$(1)_DESTDIR)/$$(1)$$$$(CO
NFIG_DST_$(1))
dst/$$$$(MY_$(1)_$$(1)_DESTDIR)/$$(1)$$$$(CONFIG_DST_$(1)): lib/$(1)/$$
(1)$$$$(CONFIG_LIB_$(1))
    -ar r lib/$(1)/libs/$$(1).a
    install -d $$$$(dir $$$$@)
    $$$$(strip libtool --mode=install install   \
        lib/$(1)/$$(1)$$$$(CONFIG_LIB_$(1)) \
        $$$$(shell pwd)/$$$$@)

override INSTALL_TARGETS += $(DESTDIR)/$$$$(MY_$(1)_$$(1)_DESTDIR)/$$(1
)$$$$(CONFIG_DST_$(1))
$(DESTDIR)/$$$$(MY_$(1)_$$(1)_DESTDIR)/$$(1)$$$$(CONFIG_DST_$(1)): dst/
$$$$(MY_$(1)_$$(1)_DESTDIR)/$$(1)$$$$(CONFIG_DST_$(1))
    install -d $$$$(dir $$$$@)
    $$$$(strip install dst/$$$$(MY_$(1)_$$(1)_DESTDIR)/$$(1)$$$$(CONFIG
_DST_$(1)) $$$$@)
    libtool --mode=finish $$$$(dir $$$$@)

$(endef__)
endef
$(foreach type,$(TYPES),$(eval $(call FUNC_BUILD_DEF,$(type))))

# Intermediate object compilers
override define FUNC_BUILD_OBJ
# $(1) == C file (no src/)
# $(2) == Parent

$(call FUNC_GEN_REAL_VAR,OBJ,$(subst /,_,$(1:%.c=%)),DEPFLAGS,$(2))
$(call FUNC_GEN_REAL_VAR,OBJ,$(subst /,_,$(1:%.c=%)),CPPFLAGS,$(2))
                                                    115,1         28%
```

A snippet of the KDUAL makefile.