

[PATCH 6/6] remove remaining errno and __KERNEL_SYSCALLS__ references

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2006-08/msg10248.html>

- *From:* Arnd Bergmann <arnd@xxxxxxxx>
 - *Date:* Wed, 30 Aug 2006 14:44:02 +0200
-

The last in-kernel user of errno is gone, so we should remove the definition and everything referring to it. This also removes the now-unused lib/execve.c file that was introduced earlier.

Also remove every trace of __KERNEL_SYSCALLS__ that still remained in the kernel.

Signed-off-by: Arnd Bergmann <arnd@xxxxxxxx>

Index: linux-cg/include/asm-cris/unistd.h

```
=====
--- linux-cg.orig/include/asm-cris/unistd.h 2006-08-30 14:38:49.000000000 +0200
+++ linux-cg/include/asm-cris/unistd.h 2006-08-30 14:43:16.000000000 +0200
@@ -324,67 +324,6 @@
#define __ARCH_WANT_SYS_RT_SIGACTION
#endif

-#ifdef __KERNEL_SYSCALLS__
-
-#include <linux/compiler.h>
-#include <linux/types.h>
-#include <linux/linkage.h>
-
-/*
- * we need this inline - forking from kernel space will result
- * in NO COPY ON WRITE (!!), until an execve is executed. This
- * is no problem, but for the stack. This is handled by not letting
- * main() use the stack at all after fork(). Thus, no function
- * calls - which means inline code for fork too, as otherwise we
- * would use the stack upon exit from 'fork()'.
- *
- * Actually only pause and fork are needed inline, so that there
- * won't be any messing with the stack from main(), but we define
- * some others too.
- */
-#define __NR__exit __NR_exit
-static inline _syscall0(pid_t,setsid)
-static inline _syscall3(int,write,int,fd,const char *,buf,off_t,count)
```

[PATCH 6/6] remove remaining errno and __KERNEL_SYSCALLS__ references

```
-static inline _syscall3(int,read,int,fd,char *,buf,off_t,count)
-static inline _syscall3(off_t,lseek,int,fd,off_t,offset,int,count)
-static inline _syscall1(int,dup,int,fd)
-static inline _syscall3(int,execve,const char *,file,char **,argv,char **,envp)
-static inline _syscall3(int,open,const char *,file,int,flag,int,mode)
-static inline _syscall1(int,close,int,fd)
-
-struct pt_regs;
-asmlinkage long sys_mmap2(
- unsigned long addr, unsigned long len,
- unsigned long prot, unsigned long flags,
- unsigned long fd, unsigned long pgoff);
-asmlinkage int sys_execve(const char *fname, char **argv, char **envp,
- long r13, long mof, long srp, struct pt_regs *regs);
-asmlinkage int sys_clone(unsigned long newusp, unsigned long flags,
- int* parent_tid, int* child_tid, long mof, long srp,
- struct pt_regs *regs);
-asmlinkage int sys_fork(long r10, long r11, long r12, long r13,
- long mof, long srp, struct pt_regs *regs);
-asmlinkage int sys_vfork(long r10, long r11, long r12, long r13,
- long mof, long srp, struct pt_regs *regs);
-asmlinkage int sys_pipe(unsigned long __user *fildes);
-struct sigaction;
-asmlinkage long sys_rt_sigaction(int sig,
- const struct sigaction __user *act,
- struct sigaction __user *oact,
- size_t sigsetsize);
-
-/*
- * Since we define it "external", it collides with the built-in
- * definition, which has the "noreturn" attribute and will cause
- * complaints. We don't want to use -fno-builtin, so just use a
- * different name when in the kernel.
- */
-#define _exit kernel_syscall_exit
-static inline _syscall1(int,_exit,int,exitcode)
-static inline _syscall3(pid_t,waitpid,pid_t,pid,int *,wait_stat,int,options)
-#endif /* __KERNEL_SYSCALLS__ */
-
-
-/*
- * "Conditional" syscalls
- */
Index: linux-cg/include/asm-frv/unistd.h
=====
--- linux-cg.orig/include/asm-frv/unistd.h 2006-08-30 14:38:49.000000000 +0200
+++ linux-cg/include/asm-frv/unistd.h 2006-08-30 14:43:16.000000000 +0200
@@ -439,31 +439,6 @@
__syscall_return(type, __sc0); \
}
```

[PATCH 6/6] remove remaining errno and __KERNEL_SYSCALLS__ references

```
-
-#ifdef __KERNEL_SYSCALLS__
-
-#include <linux/compiler.h>
-#include <linux/types.h>
-#include <linux/linkage.h>
-#include <asm/ptrace.h>
-
-/*
- * we need this inline – forking from kernel space will result
- * in NO COPY ON WRITE (!!!), until an execve is executed. This
- * is no problem, but for the stack. This is handled by not letting
- * main() use the stack at all after fork(). Thus, no function
- * calls – which means inline code for fork too, as otherwise we
- * would use the stack upon exit from 'fork()'.
- *
- * Actually only pause and fork are needed inline, so that there
- * won't be any messing with the stack from main(), but we define
- * some others too.
- */
-#define __NR_exit __NR_exit
-static inline _syscall3(int,execve,const char *,file,char **,argv,char **,envp)
-
-#endif /* __KERNEL_SYSCALLS__ */
-
#define __ARCH_WANT_IPC_PARSE_VERSION
/* #define __ARCH_WANT_OLD_READDIR */
#define __ARCH_WANT_OLD_STAT
Index: linux-cg/include/asm-h8300/unistd.h
=====
--- linux-cg.orig/include/asm-h8300/unistd.h 2006-08-30 14:38:49.000000000 +0200
+++ linux-cg/include/asm-h8300/unistd.h 2006-08-30 14:43:16.000000000 +0200
@@ -485,57 +485,6 @@
#define __ARCH_WANT_SYS_SIGPROCMASK
#define __ARCH_WANT_SYS_RT_SIGACTION

-#ifdef __KERNEL_SYSCALLS__
-
-#include <linux/compiler.h>
-#include <linux/types.h>
-
-/*
- * we need this inline – forking from kernel space will result
- * in NO COPY ON WRITE (!!!), until an execve is executed. This
- * is no problem, but for the stack. This is handled by not letting
- * main() use the stack at all after fork(). Thus, no function
- * calls – which means inline code for fork too, as otherwise we
- * would use the stack upon exit from 'fork()'.
- *
- * Actually only pause and fork are needed inline, so that there
- * won't be any messing with the stack from main(), but we define
```

[PATCH 6/6] remove remaining errno and __KERNEL_SYSCALLS__ references

```
- * some others too.
- */
-#define __NR__exit __NR__exit
-static inline _syscall0(int,pause)
-static inline _syscall0(int,sync)
-static inline _syscall0(pid_t,setsid)
-static inline _syscall3(int,write,int,fd,const char *,buf,off_t,count)
-static inline _syscall3(int,read,int,fd,char *,buf,off_t,count)
-static inline _syscall3(off_t,lseek,int,fd,off_t,offset,int,count)
-static inline _syscall1(int,dup,int,fd)
-static inline _syscall3(int,execve,const char *,file,char **,argv,char **,envp)
-static inline _syscall3(int,open,const char *,file,int,flag,int,mode)
-static inline _syscall1(int,close,int,fd)
-static inline _syscall1(int,_exit,int,exitcode)
-static inline _syscall3(pid_t,waitpid,pid_t,pid,int *,wait_stat,int,options)
-static inline _syscall1(int,delete_module,const char *,name)
-
-static inline pid_t wait(int * wait_stat)
-{
- return waitpid(-1,wait_stat,0);
-}
-
-asmlinkage long sys_mmap2(unsigned long addr, unsigned long len,
- unsigned long prot, unsigned long flags,
- unsigned long fd, unsigned long pgoff);
-asmlinkage int sys_execve(char *name, char **argv, char **envp,
- int dummy, ...);
-asmlinkage int sys_pipe(unsigned long *fildes);
-struct sigaction;
-asmlinkage long sys_rt_sigaction(int sig,
- const struct sigaction __user *act,
- struct sigaction __user *oact,
- size_t sigsetsize);
-
-#endif /* __KERNEL_SYSCALLS__ */
-
/*
* "Conditional" syscalls
*/
Index: linux-cg/include/asm-m32r/unistd.h
=====
--- linux-cg.orig/include/asm-m32r/unistd.h 2006-08-30 14:38:49.000000000 +0200
+++ linux-cg/include/asm-m32r/unistd.h 2006-08-30 14:43:16.000000000 +0200
@@ -423,43 +423,6 @@
#define __ARCH_WANT_SYS_OLDUMOUNT
#define __ARCH_WANT_SYS_RT_SIGACTION

-#ifdef __KERNEL_SYSCALLS__
-
-#include <linux/compiler.h>
-#include <linux/types.h>
```

[PATCH 6/6] remove remaining errno and __KERNEL_SYSCALLS__ references

```

-#include <linux/linkage.h>
-#include <asm/ptrace.h>
-
-/*
- * we need this inline – forking from kernel space will result
- * in NO COPY ON WRITE (!!!), until an execve is executed. This
- * is no problem, but for the stack. This is handled by not letting
- * main() use the stack at all after fork(). Thus, no function
- * calls – which means inline code for fork too, as otherwise we
- * would use the stack upon exit from 'fork()'.
- *
- * Actually only pause and fork are needed inline, so that there
- * won't be any messing with the stack from main(), but we define
- * some others too.
- */
-static __inline__ _syscall3(int,execve,const char *,file,char **,argv,char **,envp)
-
-asmlinkage long sys_mmap2(unsigned long addr, unsigned long len,
- unsigned long prot, unsigned long flags,
- unsigned long fd, unsigned long pgoff);
-asmlinkage int sys_execve(struct pt_regs regs);
-asmlinkage int sys_clone(struct pt_regs regs);
-asmlinkage int sys_fork(struct pt_regs regs);
-asmlinkage int sys_vfork(struct pt_regs regs);
-asmlinkage int sys_pipe(unsigned long __user *fildes);
-struct sigaction;
-asmlinkage long sys_rt_sigaction(int sig,
- const struct sigaction __user *act,
- struct sigaction __user *oact,
- size_t sigsetsize);
-
-#endif /* __KERNEL_SYSCALLS__ */
-
-/*
- * "Conditional" syscalls
- *
-Index: linux-cg/include/asm-m68k/unistd.h
-=====
--- linux-cg.orig/include/asm-m68k/unistd.h 2006-08-30 14:38:49.000000000 +0200
+++ linux-cg/include/asm-m68k/unistd.h 2006-08-30 14:43:16.000000000 +0200
@@ -408,12 +408,6 @@
#define __ARCH_WANT_SYS_SIGPROCMASK
#define __ARCH_WANT_SYS_RT_SIGACTION
-#ifdef __KERNEL_SYSCALLS__
-
-static inline _syscall3(int,execve,const char *,file,char **,argv,char **,envp)
-
-#endif /* __KERNEL_SYSCALLS__ */
-
-/*
```

[PATCH 6/6] remove remaining errno and __KERNEL_SYSCALLS__ references

* "Conditional" syscalls

*

Index: linux-cg/include/asm-m68knommu/unistd.h

```
=====
--- linux-cg.orig/include/asm-m68knommu/unistd.h 2006-08-30 14:38:49.000000000 +0200
+++ linux-cg/include/asm-m68knommu/unistd.h 2006-08-30 14:43:16.000000000 +0200
@@ -462,61 +462,6 @@
#define __ARCH_WANT_SYS_SIGPROC_MASK
#define __ARCH_WANT_SYS_RT_SIGACTION

-#ifdef __KERNEL_SYSCALLS__
-
-#include <linux/compiler.h>
-#include <linux/interrupt.h>
-#include <linux/types.h>
-
-/*
- * we need this inline - forking from kernel space will result
- * in NO COPY ON WRITE (!!), until an execve is executed. This
- * is no problem, but for the stack. This is handled by not letting
- * main() use the stack at all after fork(). Thus, no function
- * calls - which means inline code for fork too, as otherwise we
- * would use the stack upon exit from 'fork()'.
- *
- * Actually only pause and fork are needed inline, so that there
- * won't be any messing with the stack from main(), but we define
- * some others too.
- */
-#define __NR__exit __NR__exit
-static inline _syscall0(int,pause)
-static inline _syscall0(int, sync)
-static inline _syscall0(pid_t, setsid)
-static inline _syscall3(int, write, int, fd, const char *, buf, off_t, count)
-static inline _syscall3(int, read, int, fd, char *, buf, off_t, count)
-static inline _syscall3(off_t, lseek, int, fd, off_t, offset, int, count)
-static inline _syscall1(int, dup, int, fd)
-static inline _syscall3(int, execve, const char *, file, char **, argv, char **, envp)
-static inline _syscall3(int, open, const char *, file, int, flag, int, mode)
-static inline _syscall1(int, close, int, fd)
-static inline _syscall1(int, _exit, int, exitcode)
-static inline _syscall3(pid_t, waitpid, pid_t, pid, int *, wait_stat, int, options)
-static inline _syscall1(int, delete_module, const char *, name)
-
-static inline pid_t wait(int * wait_stat)
-{
- return waitpid(-1, wait_stat, 0);
-}
-asmlinkage long sys_mmap2(unsigned long addr, unsigned long len,
- unsigned long prot, unsigned long flags,
- unsigned long fd, unsigned long pgoff);
-asmlinkage int sys_execve(char *name, char **argv, char **envp);
```

[PATCH 6/6] remove remaining errno and __KERNEL_SYSCALLS__ references

[PATCH 6/6] remove remaining errno and __KERNEL_SYSCALLS__ references

```
-asmlinkage int sys_pipe(unsigned long *fildes);
-struct pt_regs;
-int sys_request_irq(unsigned int,
- irqreturn_t (*)(int, void *, struct pt_regs *),
- unsigned long, const char *, void *);
-void sys_free_irq(unsigned int, void *);
-struct sigaction;
-asmlinkage long sys_rt_sigaction(int sig,
- const struct sigaction __user *act,
- struct sigaction __user *oact,
- size_t sigsetsize);
-
-#endif /* __KERNEL_SYSCALLS__ */
-
/*
 * "Conditional" syscalls
 */
Index: linux-cg/include/asm-mips/unistd.h
=====
--- linux-cg.orig/include/asm-mips/unistd.h 2006-08-30 14:38:49.000000000 +0200
+++ linux-cg/include/asm-mips/unistd.h 2006-08-30 14:43:16.000000000 +0200
@@ -1206,45 +1206,6 @@
# define __ARCH_WANT_COMPAT_SYS_TIME
# endif

-#ifdef __KERNEL_SYSCALLS__
-
-#include <linux/compiler.h>
-#include <linux/types.h>
-#include <linux/linkage.h>
-#include <asm/ptrace.h>
-#include <asm/sim.h>
-
-/*
- * we need this inline - forking from kernel space will result
- * in NO COPY ON WRITE (!!!), until an execve is executed. This
- * is no problem, but for the stack. This is handled by not letting
- * main() use the stack at all after fork(). Thus, no function
- * calls - which means inline code for fork too, as otherwise we
- * would use the stack upon exit from 'fork()'.
- */
-
- * Actually only pause and fork are needed inline, so that there
- * won't be any messing with the stack from main(), but we define
- * some others too.
- */
-static inline _syscall3(int,execve,const char *,file,char **,argv,char **,envp)
-
-asmlinkage unsigned long sys_mmap(
- unsigned long addr, size_t len,
- int prot, int flags,
- int fd, off_t offset);
```

[PATCH 6/6] remove remaining errno and __KERNEL_SYSCALLS__ references

```
-asmlinkage long sys_mmap2(
- unsigned long addr, unsigned long len,
- unsigned long prot, unsigned long flags,
- unsigned long fd, unsigned long pgoff);
-asmlinkage int sys_execve(nabi_no_regargs struct pt_regs regs);
-asmlinkage int sys_pipe(nabi_no_regargs struct pt_regs regs);
-struct sigaction;
-asmlinkage long sys_rt_sigaction(int sig,
- const struct sigaction __user *act,
- struct sigaction __user *oact,
- size_t sigsetsize);
-
-#endif /* __KERNEL_SYSCALLS__ */
#endif /* !__ASSEMBLY__ */

/*
Index: linux-cg/include/asm-s390/unistd.h
=====
--- linux-cg.orig/include/asm-s390/unistd.h 2006-08-30 14:38:49.000000000 +0200
+++ linux-cg/include/asm-s390/unistd.h 2006-08-30 14:43:16.000000000 +0200
@@ -573,57 +573,6 @@
# define __ARCH_WANT_COMPAT_SYS_RT_SIGSUSPEND
# endif

-#ifdef __KERNEL_SYSCALLS__
-
-#include <linux/compiler.h>
-#include <linux/types.h>
-#include <asm/ptrace.h>
-#include <asm/stat.h>
-#include <linux/syscalls.h>
-
-/*
- * we need this inline - forking from kernel space will result
- * in NO COPY ON WRITE (!!!), until an execve is executed. This
- * is no problem, but for the stack. This is handled by not letting
- * main() use the stack at all after fork(). Thus, no function
- * calls - which means inline code for fork too, as otherwise we
- * would use the stack upon exit from 'fork()'.
- *
- * Actually only pause and fork are needed inline, so that there
- * won't be any messing with the stack from main(), but we define
- * some others too.
- */
-#define __NR_exit __NR_exit
-static inline _syscall0(pid_t,setsid)
-static inline _syscall3(int,write,int,fd,const char *,buf,off_t,count)
-static inline _syscall3(int,read,int,fd,char *,buf,off_t,count)
-static inline _syscall3(off_t,lseek,int,fd,off_t,offset,int,count)
-static inline _syscall1(int,dup,int,fd)
-static inline _syscall3(int,execve,const char *,file,char **,argv,char **,envp)
```

[PATCH 6/6] remove remaining errno and __KERNEL_SYSCALLS__ references

```
-static inline _syscall3(int,open,const char *,file,int,flag,int,mode)
-static inline _syscall1(int,close,int,fd)
-static inline _syscall2(long,stat,char *,filename,struct stat *,statbuf)
-
-static inline pid_t waitpid(int pid, int *wait_stat, int flags)
- {
- return sys_wait4(pid, wait_stat, flags, NULL);
- }
-struct mmap_arg_struct;
-asmlinkage long sys_mmap2(struct mmap_arg_struct __user *arg);
-
-asmlinkage long sys_execve(struct pt_regs regs);
-asmlinkage long sys_clone(struct pt_regs regs);
-asmlinkage long sys_fork(struct pt_regs regs);
-asmlinkage long sys_vfork(struct pt_regs regs);
-asmlinkage long sys_pipe(unsigned long __user *fildes);
-struct sigaction;
-asmlinkage long sys_rt_sigaction(int sig,
- const struct sigaction __user *act,
- struct sigaction __user *oact,
- size_t sigsetsize);
-
-#endif /* __KERNEL_SYSCALLS__ */
-
/*
* "Conditional" syscalls
*
Index: linux-cg/include/asm-sh/unistd.h
=====
--- linux-cg.orig/include/asm-sh/unistd.h 2006-08-30 14:38:49.000000000 +0200
+++ linux-cg/include/asm-sh/unistd.h 2006-08-30 14:43:16.000000000 +0200
@@ -445,76 +445,6 @@
#define __ARCH_WANT_SYS_SIGPROC_MASK
#define __ARCH_WANT_SYS_RT_SIGACTION
-#ifdef __KERNEL_SYSCALLS__
-
-#include <linux/compiler.h>
-#include <linux/types.h>
-#include <linux/linkage.h>
-#include <asm/ptrace.h>
-
-/*
- * we need this inline - forking from kernel space will result
- * in NO COPY ON WRITE (!!!), until an execve is executed. This
- * is no problem, but for the stack. This is handled by not letting
- * main() use the stack at all after fork(). Thus, no function
- * calls - which means inline code for fork too, as otherwise we
- * would use the stack upon exit from 'fork()'.
- *
- * Actually only pause and fork are needed inline, so that there
```

[PATCH 6/6] remove remaining errno and __KERNEL_SYSCALLS__ references

```
- * won't be any messing with the stack from main(), but we define
- * some others too.
- */
-#define __NR_exit __NR_exit
-static __inline__ _syscall0(int,pause)
-static __inline__ _syscall0(int,sync)
-static __inline__ _syscall0(pid_t,setsid)
-static __inline__ _syscall3(int,write,int,fd,const char *,buf,off_t,count)
-static __inline__ _syscall3(int,read,int,fd,char *,buf,off_t,count)
-static __inline__ _syscall3(off_t,lseek,int,fd,off_t,offset,int,count)
-static __inline__ _syscall1(int,dup,int,fd)
-static __inline__ _syscall3(int,execve,const char *,file,char **,argv,char **,envp)
-static __inline__ _syscall3(int,open,const char *,file,int,flag,int,mode)
-static __inline__ _syscall1(int,close,int,fd)
-static __inline__ _syscall3(pid_t,waitpid,pid_t,pid,int *,wait_stat,int,options)
-static __inline__ _syscall1(int,delete_module,const char *,name)
-
-static __inline__ pid_t wait(int * wait_stat)
-{
- return waitpid(-1,wait_stat,0);
-}
-
-asmlinkage long sys_mmap2(
- unsigned long addr, unsigned long len,
- unsigned long prot, unsigned long flags,
- unsigned long fd, unsigned long pgoff);
-asmlinkage int sys_execve(char *ufilename, char **uargv,
- char **uenvp, unsigned long r7,
- struct pt_regs regs);
-asmlinkage int sys_clone(unsigned long clone_flags, unsigned long newsp,
- unsigned long parent_tidptr,
- unsigned long child_tidptr,
- struct pt_regs regs);
-asmlinkage int sys_fork(unsigned long r4, unsigned long r5,
- unsigned long r6, unsigned long r7,
- struct pt_regs regs);
-asmlinkage int sys_vfork(unsigned long r4, unsigned long r5,
- unsigned long r6, unsigned long r7,
- struct pt_regs regs);
-asmlinkage int sys_pipe(unsigned long r4, unsigned long r5,
- unsigned long r6, unsigned long r7,
- struct pt_regs regs);
-asmlinkage ssize_t sys_pread_wrapper(unsigned int fd, char *buf,
- size_t count, long dummy, loff_t pos);
-asmlinkage ssize_t sys_pwrite_wrapper(unsigned int fd, const char *buf,
- size_t count, long dummy, loff_t pos);
-struct sigaction;
-asmlinkage long sys_rt_sigaction(int sig,
- const struct sigaction __user *act,
- struct sigaction __user *oact,
- size_t sigsetsize);
```

```

-
-#endif /* __KERNEL_SYSCALLS__ */
-
/*
 * "Conditional" syscalls
 *
Index: linux-cg/include/asm-sh64/unistd.h
=====
--- linux-cg.orig/include/asm-sh64/unistd.h 2006-08-30 14:38:49.000000000 +0200
+++ linux-cg/include/asm-sh64/unistd.h 2006-08-30 14:43:16.000000000 +0200
@@ -511,47 +511,6 @@
#define __ARCH_WANT_SYS_SIGPROC_MASK
#define __ARCH_WANT_SYS_RT_SIGACTION

-#ifdef __KERNEL_SYSCALLS__
-
-/* Copy from sh */
-#include <linux/compiler.h>
-#include <linux/types.h>
-#include <asm/ptrace.h>
-
-/*
- * we need this inline - forking from kernel space will result
- * in NO COPY ON WRITE (!!!), until an execve is executed. This
- * is no problem, but for the stack. This is handled by not letting
- * main() use the stack at all after fork(). Thus, no function
- * calls - which means inline code for fork too, as otherwise we
- * would use the stack upon exit from 'fork()'.
- *
- * Actually only pause and fork are needed inline, so that there
- * won't be any messing with the stack from main(), but we define
- * some others too.
- */
-#define __NR_exit __NR_exit
-static inline _syscall0(int,pause)
-static inline _syscall1(int,setup,int,magic)
-static inline _syscall0(int,sync)
-static inline _syscall0(pid_t,setsid)
-static inline _syscall3(int,write,int,fd,const char *,buf,off_t,count)
-static inline _syscall3(int,read,int,fd,char *,buf,off_t,count)
-static inline _syscall3(off_t,lseek,int,fd,off_t,offset,int,count)
-static inline _syscall1(int,dup,int,fd)
-static inline _syscall3(int,execve,const char *,file,char **,argv,char **,envp)
-static inline _syscall3(int,open,const char *,file,int,flag,int,mode)
-static inline _syscall1(int,close,int,fd)
-static inline _syscall1(int,_exit,int,exitcode)
-static inline _syscall3(pid_t,waitpid,pid_t,pid,int *,wait_stat,int,options)
-static inline _syscall1(int,delete_module,const char *,name)
-
-static inline pid_t wait(int * wait_stat)
-
-

```

[PATCH 6/6] remove remaining errno and __KERNEL_SYSCALLS__ references

```
- return waitpid(-1,wait_stat,0);
-}
-#endif /* __KERNEL_SYSCALLS__ */
-
/*
* "Conditional" syscalls
*
Index: linux-cg/include/asm-sparc/unistd.h
=====
--- linux-cg.orig/include/asm-sparc/unistd.h 2006-08-30 14:38:49.000000000 +0200
+++ linux-cg/include/asm-sparc/unistd.h 2006-08-30 14:43:16.000000000 +0200
@@ -478,53 +478,6 @@
#define __ARCH_WANT_SYS_SIGPROC_MASK
#define __ARCH_WANT_SYS_RT_SIGSUSPEND

-#ifdef __KERNEL_SYSCALLS__
-
-#include <linux/compiler.h>
-#include <linux/types.h>
-
-/*
- * we need this inline - forking from kernel space will result
- * in NO COPY ON WRITE (!!!), until an execve is executed. This
- * is no problem, but for the stack. This is handled by not letting
- * main() use the stack at all after fork(). Thus, no function
- * calls - which means inline code for fork too, as otherwise we
- * would use the stack upon exit from 'fork()'.
- *
- * Actually only pause and fork are needed inline, so that there
- * won't be any messing with the stack from main(), but we define
- * some others too.
- */
-#define __NR_exit __NR_exit
-static __inline__ _syscall0(pid_t,setsid)
-static __inline__ _syscall3(int,write,int,fd,__const__ char *,buf,off_t,count)
-static __inline__ _syscall3(int,read,int,fd,char *,buf,off_t,count)
-static __inline__ _syscall3(off_t,lseek,int,fd,off_t,offset,int,count)
-static __inline__ _syscall1(int,dup,int,fd)
-static __inline__ _syscall3(int,execve,__const__ char *,file,char **,argv,char **,envp)
-static __inline__ _syscall3(int,open,__const__ char *,file,int,flag,int,mode)
-static __inline__ _syscall1(int,close,int,fd)
-static __inline__ _syscall3(pid_t,waitpid,pid_t,pid,int *,wait_stat,int,options)
-
-#include <linux/linkage.h>
-
-asm linkage unsigned long sys_mmap(
- unsigned long addr, unsigned long len,
- unsigned long prot, unsigned long flags,
- unsigned long fd, unsigned long off);
-asm linkage unsigned long sys_mmap2(
- unsigned long addr, unsigned long len,
```


[PATCH 6/6] remove remaining errno and __KERNEL_SYSCALLS__ references

```
-
-asmlinkage unsigned long sys_mmap(
- unsigned long addr, unsigned long len,
- unsigned long prot, unsigned long flags,
- unsigned long fd, unsigned long off);
-struct sigaction;
-asmlinkage long sys_rt_sigaction(int sig,
- const struct sigaction __user *act,
- struct sigaction __user *oact,
- void __user *restorer,
- size_t sigsetsize);
-
-#endif /* __KERNEL_SYSCALLS__ */

/* sysconf options, for SunOS compatibility */
#define _SC_ARG_MAX 1
Index: linux-cg/include/asm-v850/unistd.h
=====
--- linux-cg.orig/include/asm-v850/unistd.h 2006-08-30 14:38:49.000000000 +0200
+++ linux-cg/include/asm-v850/unistd.h 2006-08-30 14:43:16.000000000 +0200
@@ -386,57 +386,6 @@
#define __ARCH_WANT_SYS_SIGPROC_MASK
#define __ARCH_WANT_SYS_RT_SIGACTION

-#ifdef __KERNEL_SYSCALLS__
-
-#include <linux/compiler.h>
-#include <linux/types.h>
-
-/*
- * we need this inline - forking from kernel space will result
- * in NO COPY ON WRITE (!!!), until an execve is executed. This
- * is no problem, but for the stack. This is handled by not letting
- * main() use the stack at all after fork(). Thus, no function
- * calls - which means inline code for fork too, as otherwise we
- * would use the stack upon exit from 'fork()'.
- *
- * Actually only pause and fork are needed inline, so that there
- * won't be any messing with the stack from main(), but we define
- * some others too.
- */
-#define __NR_exit __NR_exit
-extern inline _syscall0(pid_t, setuid)
-extern inline _syscall3(int, write, int, fd, const char *, buf, off_t, count)
-extern inline _syscall3(int, read, int, fd, char *, buf, off_t, count)
-extern inline _syscall3(off_t, lseek, int, fd, off_t, offset, int, count)
-extern inline _syscall1(int, dup, int, fd)
-extern inline _syscall3(int, execve, const char *, file, char **, argv, char **, envp)
-extern inline _syscall3(int, open, const char *, file, int, flag, int, mode)
-extern inline _syscall1(int, close, int, fd)
-extern inline _syscall1(int, _exit, int, exitcode)
```

[PATCH 6/6] remove remaining errno and __KERNEL_SYSCALLS__ references

```
-extern inline _syscall3(pid_t,waitpid,pid_t,pid,int *,wait_stat,int,options)
-
-extern inline pid_t wait(int * wait_stat)
- {
- return waitpid (-1, wait_stat, 0);
- }
-
-unsigned long sys_mmap(unsigned long addr, size_t len,
- unsigned long prot, unsigned long flags,
- unsigned long fd, off_t offset);
-unsigned long sys_mmap2(unsigned long addr, size_t len,
- unsigned long prot, unsigned long flags,
- unsigned long fd, unsigned long pgoff);
-struct pt_regs;
-int sys_execve (char *name, char **argv, char **envp, struct pt_regs *regs);
-int sys_pipe (int *fildes);
-struct sigaction;
-asm linkage long sys_rt_sigaction(int sig,
- const struct sigaction __user *act,
- struct sigaction __user *oact,
- size_t sigsetsize);
-
-#endif /* __KERNEL_SYSCALLS__ */
-
/*
* "Conditional" syscalls
*/
Index: linux-cg/include/asm-xtensa/unistd.h
=====
--- linux-cg.orig/include/asm-xtensa/unistd.h 2006-08-30 14:38:49.000000000 +0200
+++ linux-cg/include/asm-xtensa/unistd.h 2006-08-30 14:43:16.000000000 +0200
@@ -402,11 +402,6 @@
__syscall_return(type,__res); \
}
-
-#ifdef __KERNEL_SYSCALLS__
-static __inline__ _syscall3(int,execve,const char*,file,char**,argv,char**,envp)
-#endif
-
/*
* "Conditional" syscalls
*
Index: linux-cg/include/asm-i386/unistd.h
=====
--- linux-cg.orig/include/asm-i386/unistd.h 2006-08-30 14:38:49.000000000 +0200
+++ linux-cg/include/asm-i386/unistd.h 2006-08-30 14:39:16.000000000 +0200
@@ -449,45 +449,6 @@
#define __ARCH_WANT_SYS_RT_SIGACTION
#define __ARCH_WANT_SYS_RT_SIGSUSPEND
```

[PATCH 6/6] remove remaining errno and __KERNEL_SYSCALLS__ references

```

-#ifdef __KERNEL_SYSCALLS__
-
-#include <linux/compiler.h>
-#include <linux/types.h>
-#include <linux/linkage.h>
-#include <asm/ptrace.h>
-
-/*
- * we need this inline – forking from kernel space will result
- * in NO COPY ON WRITE (!!!), until an execve is executed. This
- * is no problem, but for the stack. This is handled by not letting
- * main() use the stack at all after fork(). Thus, no function
- * calls – which means inline code for fork too, as otherwise we
- * would use the stack upon exit from 'fork()'.
- *
- * Actually only pause and fork are needed inline, so that there
- * won't be any messing with the stack from main(), but we define
- * some others too.
- */
-#static inline _syscall3(int,execve,const char *,file,char **,argv,char **,envp)
-
-#asmlinkage int sys_modify_ldt(int func, void __user *ptr, unsigned long bytecount);
-#asmlinkage long sys_mmap2(unsigned long addr, unsigned long len,
- unsigned long prot, unsigned long flags,
- unsigned long fd, unsigned long pgoff);
-#asmlinkage int sys_execve(struct pt_regs regs);
-#asmlinkage int sys_clone(struct pt_regs regs);
-#asmlinkage int sys_fork(struct pt_regs regs);
-#asmlinkage int sys_vfork(struct pt_regs regs);
-#asmlinkage int sys_pipe(unsigned long __user *fildes);
-#asmlinkage long sys_iopl(unsigned long unused);
-#struct sigaction;
-#asmlinkage long sys_rt_sigaction(int sig,
- const struct sigaction __user *act,
- struct sigaction __user *oact,
- size_t sigsetsize);
-
-#endif /* __KERNEL_SYSCALLS__ */
-
-/*
- * "Conditional" syscalls
- *
- Index: linux-cg/arch/ia64/kernel/process.c
- =====
- --- linux-cg.orig/arch/ia64/kernel/process.c 2006-08-30 14:40:15.000000000 +0200
- +++ linux-cg/arch/ia64/kernel/process.c 2006-08-30 14:41:06.000000000 +0200
- @@ -8,8 +8,6 @@
- * 2005-10-07 Keith Owens <kaos@xxxxxxx>
- * Add notify_die() hooks.
- */
-#define __KERNEL_SYSCALLS__ /* see <asm/unistd.h> */

```

[PATCH 6/6] remove remaining errno and __KERNEL_SYSCALLS__ references

```
-
#include <linux/cpu.h>
#include <linux/pm.h>
#include <linux/elf.h>
Index: linux-cg/drivers/media/dvb/dvb-core/dvb_ringbuffer.c
=====
--- linux-cg.orig/drivers/media/dvb/dvb-core/dvb_ringbuffer.c 2006-08-30 14:40:15.000000000 +0200
+++ linux-cg/drivers/media/dvb/dvb-core/dvb_ringbuffer.c 2006-08-30 14:41:06.000000000 +0200
@@ -26,7 +26,6 @@

-#define __KERNEL_SYSCALLS__
#include <linux/errno.h>
#include <linux/kernel.h>
#include <linux/module.h>
Index: linux-cg/drivers/net/wireless/ipw2100.c
=====
--- linux-cg.orig/drivers/net/wireless/ipw2100.c 2006-08-30 14:40:15.000000000 +0200
+++ linux-cg/drivers/net/wireless/ipw2100.c 2006-08-30 14:41:06.000000000 +0200
@@ -150,7 +150,6 @@
#include <linux/skbuff.h>
#include <asm/uaccess.h>
#include <asm/io.h>
-#define __KERNEL_SYSCALLS__
#include <linux/fs.h>
#include <linux/mm.h>
#include <linux/slab.h>
Index: linux-cg/include/linux/unistd.h
=====
--- linux-cg.orig/include/linux/unistd.h 2006-08-30 14:40:15.000000000 +0200
+++ linux-cg/include/linux/unistd.h 2006-08-30 14:41:06.000000000 +0200
@@ -1,12 +1,8 @@
#ifdef _LINUX_UNISTD_H_
#define _LINUX_UNISTD_H_

-#ifdef __KERNEL__
-extern int errno;
-#endif
-
/*
- * Include machine specific syscallX macros
+ * Include machine specific syscall numbers
*/
#include <asm/unistd.h>

Index: linux-cg/lib/Makefile
=====
--- linux-cg.orig/lib/Makefile 2006-08-30 14:40:15.000000000 +0200
+++ linux-cg/lib/Makefile 2006-08-30 14:41:06.000000000 +0200
@@ -2,7 +2,7 @@
```

[PATCH 6/6] remove remaining errno and __KERNEL_SYSCALLS__ references

```
# Makefile for some libs needed in the kernel.  
#
```

```
-lib-y := errno.o ctype.o string.o vsprintf.o cmdline.o \  
+lib-y := ctype.o string.o vsprintf.o cmdline.o \  
bust_spinlocks.o rbtree.o radix-tree.o dump_stack.o \  
idr.o div64.o int_sqrt.o bitmap.o extable.o prio_tree.o \  
sha1.o  
@@ -33,8 +33,6 @@  
lib-y += dec_and_lock.o  
endif
```

```
-lib-y += execve.o
```

```
-  
obj-$(CONFIG_CRC_CCITT) += crc-ccitt.o  
obj-$(CONFIG_CRC16) += crc16.o  
obj-$(CONFIG_CRC32) += crc32.o  
Index: linux-cg/lib/errno.c
```

```
-----  
--- linux-cg.orig/lib/errno.c 2006-08-30 14:40:15.000000000 +0200
```

```
+++ /dev/null 1970-01-01 00:00:00.000000000 +0000
```

```
@@ -1,7 +0,0 @@
```

```
_/*
```

```
- * linux/lib/errno.c
```

```
- *
```

```
- * Copyright (C) 1991, 1992 Linus Torvalds
```

```
- */
```

```
-
```

```
-int errno;
```

```
Index: linux-cg/lib/execve.c
```

```
-----  
--- linux-cg.orig/lib/execve.c 2006-08-30 14:40:15.000000000 +0200
```

```
+++ /dev/null 1970-01-01 00:00:00.000000000 +0000
```

```
@@ -1,23 +0,0 @@
```

```
‑#include <asm/bug.h>
```

```
‑#include <asm/uaccess.h>
```

```
-
```

```
‑#define __KERNEL_SYSCALLS__
```

```
‑static int errno __attribute__((unused));
```

```
‑#include <asm/unistd.h>
```

```
-
```

```
‑#ifdef _syscall3
```

```
‑int kernel_execve (const char *filename, char *const argv[], char *const envp[])
```

```
‑ __attribute__((__weak__));
```

```
‑int kernel_execve (const char *filename, char *const argv[], char *const envp[])
```

```
‑{
```

```
‑ mm_segment_t fs = get_fs();
```

```
‑ int ret;
```

```
-
```

```
‑ WARN_ON(segment_eq(fs, USER_DS));
```

```
‑ ret = execve(filename, (char **)argv, (char **)envp);
```

[PATCH 6/6] remove remaining errno and __KERNEL_SYSCALLS__ references

[PATCH 6/6] remove remaining errno and __KERNEL_SYSCALLS__ references

```
- if (ret)
- ret = -errno;
-
- return ret;
-}
-#endif
```

--

-

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in
the body of a message to majordomo@xxxxxxxxxxxxxxxxx

More majordomo info at <http://vger.kernel.org/majordomo-info.html>

Please read the FAQ at <http://www.tux.org/lkml/>