

# [PATCH 6/7] remove all remaining \_syscallX macros

---

*Source:* <http://linux.derkeiler.com/Mailing-Lists/Kernel/2006-08/msg09398.html>

---

- *From:* Arnd Bergmann <[arnd@xxxxxxxx](mailto:arnd@xxxxxxxx)>
  - *Date:* Sun, 27 Aug 2006 23:47:40 +0200
- 

The \_syscallX macros were originally used by libc implementations, but now there is no libc that can be built against an up-to-date kernel and relies on them.

The only users of these macros are the \_\_KERNEL\_SYSCALLS\_, which are now gone as well, after execve has been removed from the kernel.

Signed-off-by: Arnd Bergmann <[arnd@xxxxxxxx](mailto:arnd@xxxxxxxx)>

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

```
=====
--- linux-cg.orig/include/asm-cris/unistd.h 2006-08-27 21:36:06.000000000 +0200
+++ linux-cg/include/asm-cris/unistd.h 2006-08-27 21:36:52.000000000 +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)
-static inline _syscall3(int,read,int,fd,char *,buf,off_t,count)
```

[PATCH 6/7] remove all remaining \_syscallX macros

```
-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 newsp, 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 2006-08-27 21:36:06.000000000 +0200
+++ linux-cg/include/asm-frv/unistd.h 2006-08-27 21:36:52.000000000 +0200
@@ -321,149 +321,6 @@
-#define NR_syscalls 310
-
-/*
```

## [PATCH 6/7] remove all remaining \_syscallX macros

```
- * process the return value of a syscall, consigning it to one of two possible fates
- * - user-visible error numbers are in the range -1 --4095: see <asm-frv/errno.h>
- */
-#undef __syscall_return
-#define __syscall_return(type, res) \
-do { \
- unsigned long __sr2 = (res); \
- if (__builtin_expect(__sr2 >= (unsigned long)(-4095), 0)) { \
- errno = (-__sr2); \
- __sr2 = ~0UL; \
- } \
- return (type) __sr2; \
-} while (0)
-
-/* XXX - _foo needs to be __foo, while __NR_bar could be _NR_bar. */
-
-#undef _syscall0
-#define _syscall0(type,name) \
-type name(void) \
-{ \
- register unsigned long __scnum __asm__ ("gr7") = (__NR_##name); \
- register unsigned long __sc0 __asm__ ("gr8"); \
- __asm__ __volatile__ ("tira gr0,#0" \
- : "=r" (__sc0) \
- : "r" (__scnum)); \
- __syscall_return(type, __sc0); \
-}
-
-#undef _syscall1
-#define _syscall1(type,name,type1,arg1) \
-type name(type1 arg1) \
-{ \
- register unsigned long __scnum __asm__ ("gr7") = (__NR_##name); \
- register unsigned long __sc0 __asm__ ("gr8") = (unsigned long) arg1; \
- __asm__ __volatile__ ("tira gr0,#0" \
- : "+r" (__sc0) \
- : "r" (__scnum)); \
- __syscall_return(type, __sc0); \
-}
-
-#undef _syscall2
-#define _syscall2(type,name,type1,arg1,type2,arg2) \
-type name(type1 arg1,type2 arg2) \
-{ \
- register unsigned long __scnum __asm__ ("gr7") = (__NR_##name); \
- register unsigned long __sc0 __asm__ ("gr8") = (unsigned long) arg1; \
- register unsigned long __sc1 __asm__ ("gr9") = (unsigned long) arg2; \
- __asm__ __volatile__ ("tira gr0,#0" \
- : "+r" (__sc0) \
- : "r" (__scnum), "r" (__sc1)); \
- __syscall_return(type, __sc0); \
```

[PATCH 6/7] remove all remaining \_syscallX macros

```
-}
-
-#undef _syscall3
-#define _syscall3(type,name,type1,arg1,type2,arg2,type3,arg3) \
-type name(type1 arg1,type2 arg2,type3 arg3) \
-{ \
- register unsigned long __scnum __asm__ ("gr7") = (__NR_##name); \
- register unsigned long __sc0 __asm__ ("gr8") = (unsigned long) arg1; \
- register unsigned long __sc1 __asm__ ("gr9") = (unsigned long) arg2; \
- register unsigned long __sc2 __asm__ ("gr10") = (unsigned long) arg3; \
- __asm__ __volatile__ ("tira gr0,#0" \
- : "+r" (__sc0) \
- : "r" (__scnum), "r" (__sc1), "r" (__sc2)); \
- __syscall_return(type, __sc0); \
-}
-
-#undef _syscall4
-#define _syscall4(type,name,type1,arg1,type2,arg2,type3,arg3,type4,arg4) \
-type name (type1 arg1, type2 arg2, type3 arg3, type4 arg4) \
-{ \
- register unsigned long __scnum __asm__ ("gr7") = (__NR_##name); \
- register unsigned long __sc0 __asm__ ("gr8") = (unsigned long) arg1; \
- register unsigned long __sc1 __asm__ ("gr9") = (unsigned long) arg2; \
- register unsigned long __sc2 __asm__ ("gr10") = (unsigned long) arg3; \
- register unsigned long __sc3 __asm__ ("gr11") = (unsigned long) arg4; \
- __asm__ __volatile__ ("tira gr0,#0" \
- : "+r" (__sc0) \
- : "r" (__scnum), "r" (__sc1), "r" (__sc2), "r" (__sc3)); \
- __syscall_return(type, __sc0); \
-}
-
-#undef _syscall5
-#define _syscall5(type,name,type1,arg1,type2,arg2,type3,arg3,type4,arg4,type5,arg5) \
-type name (type1 arg1, type2 arg2, type3 arg3, type4 arg4, type5 arg5) \
-{ \
- register unsigned long __scnum __asm__ ("gr7") = (__NR_##name); \
- register unsigned long __sc0 __asm__ ("gr8") = (unsigned long) arg1; \
- register unsigned long __sc1 __asm__ ("gr9") = (unsigned long) arg2; \
- register unsigned long __sc2 __asm__ ("gr10") = (unsigned long) arg3; \
- register unsigned long __sc3 __asm__ ("gr11") = (unsigned long) arg4; \
- register unsigned long __sc4 __asm__ ("gr12") = (unsigned long) arg5; \
- __asm__ __volatile__ ("tira gr0,#0" \
- : "+r" (__sc0) \
- : "r" (__sc1), "r" (__sc2), \
- "r" (__sc3), "r" (__sc4)); \
- __syscall_return(type, __sc0); \
-}
-
-#undef _syscall6
-#define _syscall6(type,name,type1,arg1,type2,arg2,type3,arg3,type4,arg4,type5,arg5, type6, arg6) \
-type name (type1 arg1, type2 arg2, type3 arg3, type4 arg4, type5 arg5, type6 arg6) \
```

[PATCH 6/7] remove all remaining \_syscallX macros

```

- { \
- register unsigned long __scnum __asm__ ("gr7") = (__NR_##name); \
- register unsigned long __sc0 __asm__ ("gr8") = (unsigned long) arg1; \
- register unsigned long __sc1 __asm__ ("gr9") = (unsigned long) arg2; \
- register unsigned long __sc2 __asm__ ("gr10") = (unsigned long) arg3; \
- register unsigned long __sc3 __asm__ ("gr11") = (unsigned long) arg4; \
- register unsigned long __sc4 __asm__ ("gr12") = (unsigned long) arg5; \
- register unsigned long __sc5 __asm__ ("gr13") = (unsigned long) arg6; \
- __asm__ __volatile__ ("tira gr0,#0" \
- : "+r" (__sc0) \
- : "r" (__scnum), "r" (__sc1), "r" (__sc2), \
- "r" (__sc3), "r" (__sc4), "r" (__sc5)); \
- __syscall_return(type, __sc0); \
- }
-
-
-#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 2006-08-27 21:36:06.000000000 +0200
+++ linux-cg/include/asm-h8300/unistd.h 2006-08-27 21:36:52.000000000 +0200
@@ -296,172 +296,6 @@
-#define NR_syscalls 289

```

[PATCH 6/7] remove all remaining \_syscallX macros

```
-
-/* user-visible error numbers are in the range -1 --122: see
-<asm-m68k/errno.h> */
-
-#define __syscall_return(type, res) \
-do { \
- if ((unsigned long)(res) >= (unsigned long)(-125)) { \
- /* avoid using res which is declared to be in register d0; \
- errno might expand to a function call and clobber it. */ \
- int __err = -(res); \
- errno = __err; \
- res = -1; \
- } \
- return (type) (res); \
-} while (0)
-
-#define _syscall0(type, name) \
-type name(void) \
-{ \
- register long __res __asm__("er0"); \
- __asm__ __volatile__ ("mov.l %1,er0\n\t" \
- "trapa #0\n\t" \
- : "=r" (__res) \
- : "g" (__NR_##name) \
- : "cc", "memory"); \
- __syscall_return(type, __res); \
-}
-
-#define _syscall1(type, name, atype, a) \
-type name(atype a) \
-{ \
- register long __res __asm__("er0"); \
- register long _a __asm__("er1"); \
- _a = (long)a; \
- __asm__ __volatile__ ("mov.l %1,er0\n\t" \
- "trapa #0\n\t" \
- : "=r" (__res) \
- : "g" (__NR_##name), \
- "g" (_a) \
- : "cc", "memory"); \
- __syscall_return(type, __res); \
-}
-
-#define _syscall2(type, name, atype, a, btype, b) \
-type name(atype a, btype b) \
-{ \
- register long __res __asm__("er0"); \
- register long _a __asm__("er1"); \
- register long _b __asm__("er2"); \
- _a = (long)a; \
- _b = (long)b; \

```

[PATCH 6/7] remove all remaining \_syscallX macros

[PATCH 6/7] remove all remaining \_syscallX macros

```
- __asm__ __volatile__ ("mov.l %1,er0\n\t" \  
- "trapa #0\n\t" \  
- : "=r" (__res) \  
- : "g" (__NR_###name), \  
- "g" (_a), \  
- "g" (_b) \  
- : "cc", "memory"); \  
- __syscall_return(type, __res); \  
-}  
-  
-#define _syscall3(type, name, atype, a, btype, b, ctype, c) \  
-type name(atype a, btype b, ctype c) \  
-{ \  
- register long __res __asm__("er0"); \  
- register long _a __asm__("er1"); \  
- register long _b __asm__("er2"); \  
- register long _c __asm__("er3"); \  
- _a = (long)a; \  
- _b = (long)b; \  
- _c = (long)c; \  
- __asm__ __volatile__ ("mov.l %1,er0\n\t" \  
- "trapa #0\n\t" \  
- : "=r" (__res) \  
- : "g" (__NR_###name), \  
- "g" (_a), \  
- "g" (_b), \  
- "g" (_c) \  
- : "cc", "memory"); \  
- __syscall_return(type, __res); \  
-}  
-  
-#define _syscall4(type, name, atype, a, btype, b, \  
- ctype, c, dtype, d) \  
-type name(atype a, btype b, ctype c, dtype d) \  
-{ \  
- register long __res __asm__("er0"); \  
- register long _a __asm__("er1"); \  
- register long _b __asm__("er2"); \  
- register long _c __asm__("er3"); \  
- register long _d __asm__("er4"); \  
- _a = (long)a; \  
- _b = (long)b; \  
- _c = (long)c; \  
- _d = (long)d; \  
- __asm__ __volatile__ ("mov.l %1,er0\n\t" \  
- "trapa #0\n\t" \  
- : "=r" (__res) \  
- : "g" (__NR_###name), \  
- "g" (_a), \  
- "g" (_b), \  
- "g" (_c), \  
- "g" (_d), \  
- : "cc", "memory"); \  
- __syscall_return(type, __res); \  
-}
```

[PATCH 6/7] remove all remaining \_syscallX macros

```
- "g" (_d) \  
- : "cc", "memory"); \  
- __syscall_return(type, __res); \  
-}  
-  
-#define _syscall5(type, name, atype, a, btype, b, \  
- ctype, c, dtype, d, etype, e) \  
-type name(atype a, btype b, ctype c, dtype d, etype e) \  
-{ \  
- register long __res __asm__("er0"); \  
- register long _a __asm__("er1"); \  
- register long _b __asm__("er2"); \  
- register long _c __asm__("er3"); \  
- register long _d __asm__("er4"); \  
- register long _e __asm__("er5"); \  
- _a = (long)a; \  
- _b = (long)b; \  
- _c = (long)c; \  
- _d = (long)d; \  
- _e = (long)e; \  
- __asm__ __volatile__ ("mov.l %1,er0\n\t" \  
- "trapa #0\n\t" \  
- : "=r" (__res) \  
- : "g" (__NR_##name), \  
- "g" (_a), \  
- "g" (_b), \  
- "g" (_c), \  
- "g" (_d), \  
- "g" (_e) \  
- : "cc", "memory"); \  
- __syscall_return(type, __res); \  
-}  
-  
-#define _syscall6(type, name, atype, a, btype, b, \  
- ctype, c, dtype, d, etype, e, ftype, f) \  
-type name(atype a, btype b, ctype c, dtype d, etype e, ftype f) \  
-{ \  
- register long __res __asm__("er0"); \  
- register long _a __asm__("er1"); \  
- register long _b __asm__("er2"); \  
- register long _c __asm__("er3"); \  
- register long _d __asm__("er4"); \  
- register long _e __asm__("er5"); \  
- register long _f __asm__("er6"); \  
- _a = (long)a; \  
- _b = (long)b; \  
- _c = (long)c; \  
- _d = (long)d; \  
- _e = (long)e; \  
- _f = (long)f; \  
- __asm__ __volatile__ ("mov.l %1,er0\n\t" \  

```

[PATCH 6/7] remove all remaining \_syscallX macros

```
- "trapa #0\n\t" \  
- : "=r" (__res) \  
- : "g" (__NR_##name), \  
- "g" (_a), \  
- "g" (_b), \  
- "g" (_c), \  
- "g" (_d), \  
- "g" (_e) \  
- "g" (_f) \  
- : "cc", "memory"); \  
- __syscall_return(type, __res); \  
-}  
-  
#define __ARCH_WANT_IPC_PARSE_VERSION  
#define __ARCH_WANT_OLD_READDIR  
#define __ARCH_WANT_OLD_STAT  
@@ -485,57 +319,6 @@  
#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  
- * 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)
```

[PATCH 6/7] remove all remaining \_syscallX macros

```
-
-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-i386/unistd.h
=====
--- linux-cg.orig/include/asm-i386/unistd.h 2006-08-27 21:36:06.000000000 +0200
+++ linux-cg/include/asm-i386/unistd.h 2006-08-27 21:36:52.000000000 +0200
@@ -328,103 +328,6 @@

#define NR_syscalls 318

_/*
- * user-visible error numbers are in the range -1 --128: see
- * <asm-i386/errno.h>
- */
-#define __syscall_return(type, res) \
-do { \
- if ((unsigned long)(res) >= (unsigned long)(-(128 + 1))) { \
- errno = -(res); \
- res = -1; \
- } \
- return (type) (res); \
-} while (0)
-
-/* XXX - _foo needs to be __foo, while __NR_bar could be _NR_bar. */
-#define _syscall0(type,name) \
-type name(void) \
-{ \
- long __res; \
- __asm__ volatile ("int $0x80" \
- : "=a" (__res) \
```

[PATCH 6/7] remove all remaining \_syscallX macros

```
- : "0" (__NR_###name)); \
-__syscall_return(type,__res); \
-}
-
-#define _syscall1(type,name,type1,arg1) \
-type name(type1 arg1) \
-{ \
-long __res; \
-__asm__ volatile ("push %%ebx ; movl %2,%%ebx ; int $0x80 ; pop %%ebx" \
- : "=a" (__res) \
- : "0" (__NR_###name),"ri" ((long)(arg1)) : "memory"); \
-__syscall_return(type,__res); \
-}
-
-#define _syscall2(type,name,type1,arg1,type2,arg2) \
-type name(type1 arg1,type2 arg2) \
-{ \
-long __res; \
-__asm__ volatile ("push %%ebx ; movl %2,%%ebx ; int $0x80 ; pop %%ebx" \
- : "=a" (__res) \
- : "0" (__NR_###name),"ri" ((long)(arg1)), "c" ((long)(arg2)) \
- : "memory"); \
-__syscall_return(type,__res); \
-}
-
-#define _syscall3(type,name,type1,arg1,type2,arg2,type3,arg3) \
-type name(type1 arg1,type2 arg2,type3 arg3) \
-{ \
-long __res; \
-__asm__ volatile ("push %%ebx ; movl %2,%%ebx ; int $0x80 ; pop %%ebx" \
- : "=a" (__res) \
- : "0" (__NR_###name),"ri" ((long)(arg1)), "c" ((long)(arg2)), \
- "d" ((long)(arg3)) : "memory"); \
-__syscall_return(type,__res); \
-}
-
-#define _syscall4(type,name,type1,arg1,type2,arg2,type3,arg3,type4,arg4) \
-type name (type1 arg1, type2 arg2, type3 arg3, type4 arg4) \
-{ \
-long __res; \
-__asm__ volatile ("push %%ebx ; movl %2,%%ebx ; int $0x80 ; pop %%ebx" \
- : "=a" (__res) \
- : "0" (__NR_###name),"ri" ((long)(arg1)), "c" ((long)(arg2)), \
- "d" ((long)(arg3)), "S" ((long)(arg4)) : "memory"); \
-__syscall_return(type,__res); \
-}
-
-#define _syscall5(type,name,type1,arg1,type2,arg2,type3,arg3,type4,arg4, \
-type5,arg5) \
-type name (type1 arg1,type2 arg2,type3 arg3,type4 arg4,type5 arg5) \
-{ \
```

[PATCH 6/7] remove all remaining \_syscallX macros

```
-long __res; \
-__asm__ volatile ("push %%ebx ; movl %2,%%ebx ; movl %1,%%eax ; " \
- "int $0x80 ; pop %%ebx" \
- : "=a" (__res) \
- : "i" (__NR_###name),"ri" ((long)(arg1)),"c" ((long)(arg2)), \
- "d" ((long)(arg3)),"S" ((long)(arg4)),"D" ((long)(arg5)) \
- : "memory"); \
-__syscall_return(type,__res); \
-}
-
-#define _syscall6(type,name,type1,arg1,type2,arg2,type3,arg3,type4,arg4, \
- type5,arg5,type6,arg6) \
-type name (type1 arg1,type2 arg2,type3 arg3,type4 arg4,type5 arg5,type6 arg6) \
-{ \
-long __res; \
- struct { long __a1; long __a6; } __s = { (long)arg1, (long)arg6 }; \
-__asm__ volatile ("push %%ebp ; push %%ebx ; movl 4(%2),%%ebp ; " \
- "movl 0(%2),%%ebx ; movl %1,%%eax ; int $0x80 ; " \
- "pop %%ebx ; pop %%ebp" \
- : "=a" (__res) \
- : "i" (__NR_###name),"0" ((long>(&__s)),"c" ((long)(arg2)), \
- "d" ((long)(arg3)),"S" ((long)(arg4)),"D" ((long)(arg5)) \
- : "memory"); \
-__syscall_return(type,__res); \
-}
-
#define __ARCH_WANT_IPC_PARSE_VERSION
#define __ARCH_WANT_OLD_READDIR
#define __ARCH_WANT_OLD_STAT
@@ -449,45 +352,6 @@
#define __ARCH_WANT_SYS_RT_SIGACTION
#define __ARCH_WANT_SYS_RT_SIGSUSPEND

-#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.
```

[PATCH 6/7] remove all remaining \_syscallX macros

```

- */
-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/include/asm-m32r/unistd.h
=====
--- linux-cg.orig/include/asm-m32r/unistd.h 2006-08-27 21:36:06.000000000 +0200
+++ linux-cg/include/asm-m32r/unistd.h 2006-08-27 21:36:52.000000000 +0200
@@ -299,114 +299,6 @@

#define NR_syscalls 285

-/* user-visible error numbers are in the range -1 --124: see
- * <asm-m32r/errno.h>
- */
-
-#define __syscall_return(type, res) \
-do { \
- if ((unsigned long)(res) >= (unsigned long)(-(124 + 1))) { \
- /* Avoid using "res" which is declared to be in register r0; \
- errno might expand to a function call and clobber it. */ \
- int __err = -(res); \
- errno = __err; \
- res = -1; \
- } \
- return (type) (res); \
-} while (0)
-
-#define _syscall0(type,name) \
-type name(void) \
-{ \

```

[PATCH 6/7] remove all remaining \_syscallX macros

```
-register long __scno __asm__ ("r7") = __NR_###name; \
-register long __res __asm__ ("r0"); \
-__asm__ __volatile__ (\
-"trap #" SYSCALL_VECTOR "|| nop"\
-: "=r" (__res) \
-: "r" (__scno) \
-: "memory"); \
-__syscall_return(type,__res); \
-}
-
-#define _syscall1(type,name,type1,arg1) \
-type name(type1 arg1) \
-{ \
-register long __scno __asm__ ("r7") = __NR_###name; \
-register long __res __asm__ ("r0") = (long)(arg1); \
-__asm__ __volatile__ (\
-"trap #" SYSCALL_VECTOR "|| nop"\
-: "=r" (__res) \
-: "r" (__scno), "0" (__res) \
-: "memory"); \
-__syscall_return(type,__res); \
-}
-
-#define _syscall2(type,name,type1,arg1,type2,arg2) \
-type name(type1 arg1,type2 arg2) \
-{ \
-register long __scno __asm__ ("r7") = __NR_###name; \
-register long __arg2 __asm__ ("r1") = (long)(arg2); \
-register long __res __asm__ ("r0") = (long)(arg1); \
-__asm__ __volatile__ (\
-"trap #" SYSCALL_VECTOR "|| nop"\
-: "=r" (__res) \
-: "r" (__scno), "0" (__res), "r" (__arg2) \
-: "memory"); \
-__syscall_return(type,__res); \
-}
-
-#define _syscall3(type,name,type1,arg1,type2,arg2,type3,arg3) \
-type name(type1 arg1,type2 arg2,type3 arg3) \
-{ \
-register long __scno __asm__ ("r7") = __NR_###name; \
-register long __arg3 __asm__ ("r2") = (long)(arg3); \
-register long __arg2 __asm__ ("r1") = (long)(arg2); \
-register long __res __asm__ ("r0") = (long)(arg1); \
-__asm__ __volatile__ (\
-"trap #" SYSCALL_VECTOR "|| nop"\
-: "=r" (__res) \
-: "r" (__scno), "0" (__res), "r" (__arg2), \
-"r" (__arg3) \
-: "memory"); \
-__syscall_return(type,__res); \
```

[PATCH 6/7] remove all remaining \_syscallX macros

```

-}
-
-#define _syscall4(type,name,type1,arg1,type2,arg2,type3,arg3,type4,arg4) \
-type name(type1 arg1,type2 arg2,type3 arg3,type4 arg4) \
-{ \
-register long __scno __asm__ ("r7") = __NR_##name; \
-register long __arg4 __asm__ ("r3") = (long)(arg4); \
-register long __arg3 __asm__ ("r2") = (long)(arg3); \
-register long __arg2 __asm__ ("r1") = (long)(arg2); \
-register long __res __asm__ ("r0") = (long)(arg1); \
-__asm__ __volatile__ (\
-"trap #" SYSCALL_VECTOR "|| nop" \
-: "=r" (__res) \
-: "r" (__scno), "0" (__res), "r" (__arg2), \
-"r" (__arg3), "r" (__arg4) \
-: "memory"); \
-__syscall_return(type,__res); \
-}
-
-#define _syscall5(type,name,type1,arg1,type2,arg2,type3,arg3,type4,arg4, \
-type5,arg5) \
-type name(type1 arg1,type2 arg2,type3 arg3,type4 arg4,type5 arg5) \
-{ \
-register long __scno __asm__ ("r7") = __NR_##name; \
-register long __arg5 __asm__ ("r4") = (long)(arg5); \
-register long __arg4 __asm__ ("r3") = (long)(arg4); \
-register long __arg3 __asm__ ("r2") = (long)(arg3); \
-register long __arg2 __asm__ ("r1") = (long)(arg2); \
-register long __res __asm__ ("r0") = (long)(arg1); \
-__asm__ __volatile__ (\
-"trap #" SYSCALL_VECTOR "|| nop" \
-: "=r" (__res) \
-: "r" (__scno), "0" (__res), "r" (__arg2), \
-"r" (__arg3), "r" (__arg4), "r" (__arg5) \
-: "memory"); \
-__syscall_return(type,__res); \
-}
-
#define __ARCH_WANT_IPC_PARSE_VERSION
#define __ARCH_WANT_STAT64
#define __ARCH_WANT_SYS_ALARM
@@ -423,43 +315,6 @@
#define __ARCH_WANT_SYS_OLDUMOUNT
#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>

```

[PATCH 6/7] remove all remaining \_syscallX macros

```
-
-/*
- * 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)
-
-asm__linkage long sys_mmap2(unsigned long addr, unsigned long len,
- unsigned long prot, unsigned long flags,
- unsigned long fd, unsigned long pgoff);
-asm__linkage int sys_execve(struct pt_regs regs);
-asm__linkage int sys_clone(struct pt_regs regs);
-asm__linkage int sys_fork(struct pt_regs regs);
-asm__linkage int sys_vfork(struct pt_regs regs);
-asm__linkage int sys_pipe(unsigned long __user *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-m68k/unistd.h
-=====
- --- linux-cg.orig/include/asm-m68k/unistd.h 2006-08-27 21:36:06.000000000 +0200
- +++ linux-cg/include/asm-m68k/unistd.h 2006-08-27 21:36:52.000000000 +0200
- @@ -289,102 +289,6 @@
-
-#define NR_syscalls 282
-
-/* user-visible error numbers are in the range -1 – -124: see
- <asm-m68k/errno.h> */
-
-#define __syscall_return(type, res) \
-do { \
- if ((unsigned long)(res) >= (unsigned long)(-125)) { \
- /* avoid using res which is declared to be in register d0; \
- errno might expand to a function call and clobber it. */ \
- int __err = -(res); \
```

[PATCH 6/7] remove all remaining \_syscallX macros

[PATCH 6/7] remove all remaining \_syscallX macros

```
- errno = __err; \
- res = -1; \
- } \
- return (type) (res); \
-} while (0)
-
-#define _syscall0(type,name) \
-type name(void) \
-{ \
-register long __res __asm__ ("%d0") = __NR_##name; \
- __asm__ __volatile__ ("trap #0" \
- : "+d" (__res)); \
- __syscall_return(type,__res); \
-}
-
-#define _syscall1(type,name,atype,a) \
-type name(atype a) \
-{ \
-register long __res __asm__ ("%d0") = __NR_##name; \
-register long __a __asm__ ("%d1") = (long)(a); \
- __asm__ __volatile__ ("trap #0" \
- : "+d" (__res) \
- : "d" (__a)); \
- __syscall_return(type,__res); \
-}
-
-#define _syscall2(type,name,atype,a,btype,b) \
-type name(atype a,btype b) \
-{ \
-register long __res __asm__ ("%d0") = __NR_##name; \
-register long __a __asm__ ("%d1") = (long)(a); \
-register long __b __asm__ ("%d2") = (long)(b); \
- __asm__ __volatile__ ("trap #0" \
- : "+d" (__res) \
- : "d" (__a), "d" (__b) \
- ); \
- __syscall_return(type,__res); \
-}
-
-#define _syscall3(type,name,atype,a,btype,b,ctype,c) \
-type name(atype a,btype b,ctype c) \
-{ \
-register long __res __asm__ ("%d0") = __NR_##name; \
-register long __a __asm__ ("%d1") = (long)(a); \
-register long __b __asm__ ("%d2") = (long)(b); \
-register long __c __asm__ ("%d3") = (long)(c); \
- __asm__ __volatile__ ("trap #0" \
- : "+d" (__res) \
- : "d" (__a), "d" (__b), \
- "d" (__c) \
- ); \
-}
-
```

[PATCH 6/7] remove all remaining \_syscallX macros

```
-__syscall_return(type,__res); \
-}
-
-#define _syscall4(type,name,atype,a,btype,b,ctype,c,dtype,d) \
-type name (atype a, btype b, ctype c, dtype d) \
-{ \
-register long __res __asm__ ("%d0") = __NR_##name; \
-register long __a __asm__ ("%d1") = (long)(a); \
-register long __b __asm__ ("%d2") = (long)(b); \
-register long __c __asm__ ("%d3") = (long)(c); \
-register long __d __asm__ ("%d4") = (long)(d); \
-__asm__ __volatile__ ("trap #0" \
- : "+d" (__res) \
- : "d" (__a), "d" (__b), \
- "d" (__c), "d" (__d) \
- ); \
-__syscall_return(type,__res); \
-}
-
-#define _syscall5(type,name,atype,a,btype,b,ctype,c,dtype,d,etype,e) \
-type name (atype a,btype b,ctype c,dtype d,etype e) \
-{ \
-register long __res __asm__ ("%d0") = __NR_##name; \
-register long __a __asm__ ("%d1") = (long)(a); \
-register long __b __asm__ ("%d2") = (long)(b); \
-register long __c __asm__ ("%d3") = (long)(c); \
-register long __d __asm__ ("%d4") = (long)(d); \
-register long __e __asm__ ("%d5") = (long)(e); \
-__asm__ __volatile__ ("trap #0" \
- : "+d" (__res) \
- : "d" (__a), "d" (__b), \
- "d" (__c), "d" (__d), "d" (__e) \
- ); \
-__syscall_return(type,__res); \
-}
-
#define __ARCH_WANT_IPC_PARSE_VERSION
#define __ARCH_WANT_OLD_READDIR
#define __ARCH_WANT_OLD_STAT
@@ -408,12 +312,6 @@
#define __ARCH_WANT_SYS_SIGPROC_MASK
#define __ARCH_WANT_SYS_RT_SIGACTION

-#ifdef __KERNEL_SYSCALLS__
-
-static inline _syscall3(int,execve,const char *,file,char **,argv,char **,envp)
-
-#endif /* __KERNEL_SYSCALLS__ */
-
/*
* "Conditional" syscalls
```

[PATCH 6/7] remove all remaining \_syscallX macros

\*

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

```
=====
--- linux-cg.orig/include/asm-m68knommu/unistd.h 2006-08-27 21:36:06.000000000 +0200
+++ linux-cg/include/asm-m68knommu/unistd.h 2006-08-27 21:36:52.000000000 +0200
@@ -290,155 +290,6 @@
```

```
#define NR_syscalls 282
```

```
/* user-visible error numbers are in the range -1 --122: see
<asm-m68k/errno.h> */
```

```

-#define __syscall_return(type, res) \
-do { \
- if ((unsigned long)(res) >= (unsigned long)(-125)) { \
- /* avoid using res which is declared to be in register d0; \
- errno might expand to a function call and clobber it. */ \
- int __err = -(res); \
- errno = __err; \
- res = -1; \
- } \
- return (type) (res); \
-} while (0)
```

```

-#define _syscall0(type, name) \
-type name(void) \
-{ \
- long __res; \
- __asm__ __volatile__ ("move1 %1, %%d0\n\t" \
- "trap #0\n\t" \
- "move1 %%d0, %0" \
- : "=g" (__res) \
- : "i" (__NR_##name) \
- : "cc", "%d0"); \
- if ((unsigned long)(__res) >= (unsigned long)(-125)) { \
- errno = -__res; \
- __res = -1; \
- } \
- return (type) __res; \
-}
```

```

-#define _syscall1(type, name, atype, a) \
-type name(atype a) \
-{ \
- long __res; \
- __asm__ __volatile__ ("move1 %2, %%d1\n\t" \
- "move1 %1, %%d0\n\t" \
- "trap #0\n\t" \
- "move1 %%d0, %0" \
- : "=g" (__res) \
- : "i" (__NR_##name), \
```

[PATCH 6/7] remove all remaining \_syscallX macros

[PATCH 6/7] remove all remaining \_syscallX macros

```
- "g" ((long)a) \
- : "cc", "%d0", "%d1"); \
- if ((unsigned long)(__res) >= (unsigned long)(-125)) { \
- errno = -__res; \
- __res = -1; \
- } \
- return (type)__res; \
-}
-
-#define _syscall2(type, name, atype, a, btype, b) \
-type name(atype a, btype b) \
-{ \
- long __res; \
- __asm__ __volatile__ ("movl %3, %%d2\n\t" \
- "movl %2, %%d1\n\t" \
- "movl %1, %%d0\n\t" \
- "trap #0\n\t" \
- "movl %%d0, %0" \
- : "=g" (__res) \
- : "i" (__NR_##name), \
- "a" ((long)a), \
- "g" ((long)b) \
- : "cc", "%d0", "%d1", "%d2"); \
- if ((unsigned long)(__res) >= (unsigned long)(-125)) { \
- errno = -__res; \
- __res = -1; \
- } \
- return (type)__res; \
-}
-
-#define _syscall3(type, name, atype, a, btype, b, ctype, c) \
-type name(atype a, btype b, ctype c) \
-{ \
- long __res; \
- __asm__ __volatile__ ("movl %4, %%d3\n\t" \
- "movl %3, %%d2\n\t" \
- "movl %2, %%d1\n\t" \
- "movl %1, %%d0\n\t" \
- "trap #0\n\t" \
- "movl %%d0, %0" \
- : "=g" (__res) \
- : "i" (__NR_##name), \
- "a" ((long)a), \
- "a" ((long)b), \
- "g" ((long)c) \
- : "cc", "%d0", "%d1", "%d2", "%d3"); \
- if ((unsigned long)(__res) >= (unsigned long)(-125)) { \
- errno = -__res; \
- __res = -1; \
- } \
- return (type)__res; \
-}
```

[PATCH 6/7] remove all remaining \_syscallX macros

```
-}  
-  
-#define _syscall4(type, name, atype, a, btype, b, ctype, c, dtype, d) \  
-type name(atype a, btype b, ctype c, dtype d) \  
-{ \  
- long __res; \  
- __asm__ __volatile__ ("movel %5, %%d4\n\t" \  
- "movel %4, %%d3\n\t" \  
- "movel %3, %%d2\n\t" \  
- "movel %2, %%d1\n\t" \  
- "movel %1, %%d0\n\t" \  
- "trap #0\n\t" \  
- "movel %%d0, %0" \  
- : "=g" (__res) \  
- : "i" (__NR_###name), \  
- "a" ((long)a), \  
- "a" ((long)b), \  
- "a" ((long)c), \  
- "g" ((long)d) \  
- : "cc", "%d0", "%d1", "%d2", "%d3", \  
- "%d4"); \  
- if ((unsigned long)(__res) >= (unsigned long)(-125)) { \  
- errno = -__res; \  
- __res = -1; \  
- } \  
- return (type)__res; \  
-}  
-  
-#define _syscall5(type, name, atype, a, btype, b, ctype, c, dtype, d, etype, e) \  
-type name(atype a, btype b, ctype c, dtype d, etype e) \  
-{ \  
- long __res; \  
- __asm__ __volatile__ ("movel %6, %%d5\n\t" \  
- "movel %5, %%d4\n\t" \  
- "movel %4, %%d3\n\t" \  
- "movel %3, %%d2\n\t" \  
- "movel %2, %%d1\n\t" \  
- "movel %1, %%d0\n\t" \  
- "trap #0\n\t" \  
- "movel %%d0, %0" \  
- : "=g" (__res) \  
- : "i" (__NR_###name), \  
- "a" ((long)a), \  
- "a" ((long)b), \  
- "a" ((long)c), \  
- "a" ((long)d), \  
- "g" ((long)e) \  
- : "cc", "%d0", "%d1", "%d2", "%d3", \  
- "%d4", "%d5"); \  
- if ((unsigned long)(__res) >= (unsigned long)(-125)) { \  
- errno = -__res; \  
- }
```

[PATCH 6/7] remove all remaining \_syscallX macros

```
- __res = -1; \
- } \
- return (type)__res; \
-}
-
#define __ARCH_WANT_IPC_PARSE_VERSION
#define __ARCH_WANT_OLD_READDIR
#define __ARCH_WANT_OLD_STAT
@@ -462,61 +313,6 @@
#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,sysync)
-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);
-}
-asm linkage long sys_mmap2(unsigned long addr, unsigned long len,
- unsigned long prot, unsigned long flags,
```

[PATCH 6/7] remove all remaining \_syscallX macros

[PATCH 6/7] remove all remaining \_syscallX macros

```
- unsigned long fd, unsigned long pgoff);
- asmlinkage int sys_execve(char *name, char **argv, char **envp);
- 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 2006-08-27 21:36:06.000000000 +0200
+++ linux-cg/include/asm-mips/unistd.h 2006-08-27 21:36:52.000000000 +0200
@@ -918,268 +918,6 @@
```

```
#ifndef __ASSEMBLY__
```

```
/* XXX - _foo needs to be __foo, while __NR_bar could be _NR_bar. */
#define _syscall0(type,name) \
-type name(void) \
-{ \
- register unsigned long __a3 asm("$7"); \
- unsigned long __v0; \
- \
- __asm__ volatile ( \
- ".set\tnoreorder\n\t" \
- "li\t$2, %2\t\t\t# name "\n\t" \
- "syscall\n\t" \
- "move\t%0, $2\n\t" \
- ".set\treorder" \
- : "=&r" (__v0), "=r" (__a3) \
- : "i" (__NR_##name) \
- : "$2", "$8", "$9", "$10", "$11", "$12", "$13", "$14", "$15", "$24", \
- "memory"); \
- \
- if (__a3 == 0) \
- return (type) __v0; \
- errno = __v0; \
- return (type) -1; \
-}
-
```

[PATCH 6/7] remove all remaining \_syscallX macros

```

_/*
- * DANGER: This macro isn't usable for the pipe(2) call
- * which has a unusual return convention.
- */
-#define _syscall1(type,name,atype,a) \
-type name(atype a) \
-{ \
- register unsigned long __a0 asm("$4") = (unsigned long) a; \
- register unsigned long __a3 asm("$7"); \
- unsigned long __v0; \
- \
- __asm__ volatile ( \
- ".set\tnoreorder\n\t" \
- "li\t$2, %3\t\t\t# #name "\n\t" \
- "syscall\n\t" \
- "move\t%0, $2\n\t" \
- ".set\treorder" \
- : "=&r" (__v0), "=r" (__a3) \
- : "r" (__a0), "i" (__NR_##name) \
- : "$2", "$8", "$9", "$10", "$11", "$12", "$13", "$14", "$15", "$24", \
- "memory"); \
- \
- if (__a3 == 0) \
- return (type) __v0; \
- errno = __v0; \
- return (type) -1; \
-}
-
-#define _syscall2(type,name,atype,a,btype,b) \
-type name(atype a, btype b) \
-{ \
- register unsigned long __a0 asm("$4") = (unsigned long) a; \
- register unsigned long __a1 asm("$5") = (unsigned long) b; \
- register unsigned long __a3 asm("$7"); \
- unsigned long __v0; \
- \
- __asm__ volatile ( \
- ".set\tnoreorder\n\t" \
- "li\t$2, %4\t\t\t# #name "\n\t" \
- "syscall\n\t" \
- "move\t%0, $2\n\t" \
- ".set\treorder" \
- : "=&r" (__v0), "=r" (__a3) \
- : "r" (__a0), "r" (__a1), "i" (__NR_##name) \
- : "$2", "$8", "$9", "$10", "$11", "$12", "$13", "$14", "$15", "$24", \
- "memory"); \
- \
- if (__a3 == 0) \
- return (type) __v0; \
- errno = __v0; \
- return (type) -1; \
-}

```

[PATCH 6/7] remove all remaining \_syscallX macros

```
-}
-
-#define _syscall3(type,name,atype,a,btype,b,ctype,c) \
-type name(atype a, btype b, ctype c) \
-{ \
- register unsigned long __a0 asm("$4") = (unsigned long) a; \
- register unsigned long __a1 asm("$5") = (unsigned long) b; \
- register unsigned long __a2 asm("$6") = (unsigned long) c; \
- register unsigned long __a3 asm("$7"); \
- unsigned long __v0; \
- \
- __asm__ volatile ( \
- ".set\tnoreorder\n\t" \
- "li\t$2, %5\t\t\t# #name "\n\t" \
- "syscall\n\t" \
- "move\t%0, $2\n\t" \
- ".set\treorder" \
- : "=&r" (__v0), "=r" (__a3) \
- : "r" (__a0), "r" (__a1), "r" (__a2), "i" (__NR_##name) \
- : "$2", "$8", "$9", "$10", "$11", "$12", "$13", "$14", "$15", "$24", \
- "memory"); \
- \
- if (__a3 == 0) \
- return (type) __v0; \
- errno = __v0; \
- return (type) -1; \
-}
-
-#define _syscall4(type,name,atype,a,btype,b,ctype,c,dtype,d) \
-type name(atype a, btype b, ctype c, dtype d) \
-{ \
- register unsigned long __a0 asm("$4") = (unsigned long) a; \
- register unsigned long __a1 asm("$5") = (unsigned long) b; \
- register unsigned long __a2 asm("$6") = (unsigned long) c; \
- register unsigned long __a3 asm("$7") = (unsigned long) d; \
- unsigned long __v0; \
- \
- __asm__ volatile ( \
- ".set\tnoreorder\n\t" \
- "li\t$2, %5\t\t\t# #name "\n\t" \
- "syscall\n\t" \
- "move\t%0, $2\n\t" \
- ".set\treorder" \
- : "=&r" (__v0), "+r" (__a3) \
- : "r" (__a0), "r" (__a1), "r" (__a2), "i" (__NR_##name) \
- : "$2", "$8", "$9", "$10", "$11", "$12", "$13", "$14", "$15", "$24", \
- "memory"); \
- \
- if (__a3 == 0) \
- return (type) __v0; \
- errno = __v0; \
-}
```

[PATCH 6/7] remove all remaining \_syscallX macros

```

- return (type) -1; \
-}
-
-#if (_MIPS_SIM == _MIPS_SIM_ABI32)
-
-/*
- * Using those means your brain needs more than an oil change ;-)
- */
-
-#define _syscall5(type,name,atype,a,btype,b,ctype,c,dtype,d,etype,e) \
-type name(atype a, btype b, ctype c, dtype d, etype e) \
-{ \
- register unsigned long __a0 asm("$4") = (unsigned long) a; \
- register unsigned long __a1 asm("$5") = (unsigned long) b; \
- register unsigned long __a2 asm("$6") = (unsigned long) c; \
- register unsigned long __a3 asm("$7") = (unsigned long) d; \
- unsigned long __v0; \
- \
- __asm__ volatile ( \
- ".set\tnoreorder\n\t" \
- "lw\t$2, %6\n\t" \
- "subu\t$29, 32\n\t" \
- "sw\t$2, 16($29)\n\t" \
- "li\t$2, %5\t\t\t# " #name "\n\t" \
- "syscall\n\t" \
- "move\t%0, $2\n\t" \
- "addiu\t$29, 32\n\t" \
- ".set\treorder" \
- : "=&r" (__v0), "+r" (__a3) \
- : "r" (__a0), "r" (__a1), "r" (__a2), "i" (__NR_###name), \
- "m" ((unsigned long)e) \
- : "$2", "$8", "$9", "$10", "$11", "$12", "$13", "$14", "$15", "$24", \
- "memory"); \
- \
- if (__a3 == 0) \
- return (type) __v0; \
- errno = __v0; \
- return (type) -1; \
-}
-
-#define _syscall6(type,name,atype,a,btype,b,ctype,c,dtype,d,etype,e,ftype,f) \
-type name(atype a, btype b, ctype c, dtype d, etype e, ftype f) \
-{ \
- register unsigned long __a0 asm("$4") = (unsigned long) a; \
- register unsigned long __a1 asm("$5") = (unsigned long) b; \
- register unsigned long __a2 asm("$6") = (unsigned long) c; \
- register unsigned long __a3 asm("$7") = (unsigned long) d; \
- unsigned long __v0; \
- \
- __asm__ volatile ( \
- ".set\tnoreorder\n\t" \

```

[PATCH 6/7] remove all remaining \_syscallX macros

```

- "lw\t$2, %6\n\t" \
- "lw\t$8, %7\n\t" \
- "subu\t$29, 32\n\t" \
- "sw\t$2, 16($29)\n\t" \
- "sw\t$8, 20($29)\n\t" \
- "li\t$2, %5\t\t\t# name "\n\t" \
- "syscall\n\t" \
- "move\t%0, $2\n\t" \
- "addiu\t$29, 32\n\t" \
- ".set\treorder" \
- : "=&r" (__v0), "+r" (__a3) \
- : "r" (__a0), "r" (__a1), "r" (__a2), "i" (__NR_##name), \
- "m" ((unsigned long)e), "m" ((unsigned long)f) \
- : "$2", "$8", "$9", "$10", "$11", "$12", "$13", "$14", "$15", "$24", \
- "memory"); \
- \
- if (__a3 == 0) \
- return (type) __v0; \
- errno = __v0; \
- return (type) -1; \
-}
-
-#endif /* (_MIPS_SIM == _MIPS_SIM_ABI32) */
-
-#if (_MIPS_SIM == _MIPS_SIM_NABI32) || (_MIPS_SIM == _MIPS_SIM_ABI64)
-
-#define _syscall5(type,name,atype,a,btype,b,ctype,c,dtype,d,etype,e) \
- type name (atype a,btype b,ctype c,dtype d,etype e) \
- { \
- register unsigned long __a0 asm("$4") = (unsigned long) a; \
- register unsigned long __a1 asm("$5") = (unsigned long) b; \
- register unsigned long __a2 asm("$6") = (unsigned long) c; \
- register unsigned long __a3 asm("$7") = (unsigned long) d; \
- register unsigned long __a4 asm("$8") = (unsigned long) e; \
- unsigned long __v0; \
- \
- __asm__ volatile ( \
- ".set\tnoreorder\n\t" \
- "li\t$2, %6\t\t\t# name "\n\t" \
- "syscall\n\t" \
- "move\t%0, $2\n\t" \
- ".set\treorder" \
- : "=&r" (__v0), "+r" (__a3) \
- : "r" (__a0), "r" (__a1), "r" (__a2), "r" (__a4), "i" (__NR_##name) \
- : "$2", "$9", "$10", "$11", "$12", "$13", "$14", "$15", "$24", \
- "memory"); \
- \
- if (__a3 == 0) \
- return (type) __v0; \
- errno = __v0; \
- return (type) -1; \

```

[PATCH 6/7] remove all remaining \_syscallX macros

```
-}
-
-#define _syscall6(type,name,atype,a,btype,b,ctype,c,dtype,d,etype,e,ftype,f) \
-type name (atype a,btype b,ctype c,dtype d,etype e,ftype f) \
-{ \
- register unsigned long __a0 asm("$4") = (unsigned long) a; \
- register unsigned long __a1 asm("$5") = (unsigned long) b; \
- register unsigned long __a2 asm("$6") = (unsigned long) c; \
- register unsigned long __a3 asm("$7") = (unsigned long) d; \
- register unsigned long __a4 asm("$8") = (unsigned long) e; \
- register unsigned long __a5 asm("$9") = (unsigned long) f; \
- unsigned long __v0; \
- \
- __asm__ volatile ( \
- ".set\tnoreorder\n\t" \
- "li\t$2, %7\t\t\t# #name "\n\t" \
- "syscall\n\t" \
- "move\t%0, $2\n\t" \
- ".set\treorder" \
- : "=&r" (__v0), "+r" (__a3) \
- : "r" (__a0), "r" (__a1), "r" (__a2), "r" (__a4), "r" (__a5), \
- "i" (__NR_##name) \
- : "$2", "$9", "$10", "$11", "$12", "$13", "$14", "$15", "$24", \
- "memory"); \
- \
- if (__a3 == 0) \
- return (type) __v0; \
- errno = __v0; \
- return (type) -1; \
-}
-
-#endif /* (_MIPS_SIM == _MIPS_SIM_NABI32) || (_MIPS_SIM == _MIPS_SIM_ABI64) */
-
-
-#define __ARCH_WANT_IPC_PARSE_VERSION
-#define __ARCH_WANT_OLD_READDIR
-#define __ARCH_WANT_SYS_ALARM
-@@ -1206,45 +944,6 @@
-# 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
```

[PATCH 6/7] remove all remaining \_syscallX macros

```
- * 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);
-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 2006-08-27 21:36:06.000000000 +0200
+++ linux-cg/include/asm-s390/unistd.h 2006-08-27 21:36:52.000000000 +0200
@@ -394,158 +394,6 @@

#ifdef __KERNEL__

#define __syscall_return(type, res) \
-do { \
- if ((unsigned long)(res) >= (unsigned long)(-4095)) {\
- errno = -(res); \
- res = -1; \
- } \
- return (type) (res); \
-} while (0)

#define _svc_clobber "1", "cc", "memory"

```

[PATCH 6/7] remove all remaining \_syscallX macros

```

-#define _syscall0(type,name) \
-type name(void) { \
- register long __svcreas asm("2"); \
- long __res; \
- __asm__ __volatile__ ( \
- " .if %1 < 256\n" \
- " svc %b1\n" \
- " .else\n" \
- " la %%r1,%1\n" \
- " svc 0\n" \
- " .endif" \
- : "=d" (__svcreas) \
- : "i" (__NR_##name) \
- : _svc_clobber ); \
- __res = __svcreas; \
- __syscall_return(type,__res); \
-}
-
-#define _syscall1(type,name,type1,arg1) \
-type name(type1 arg1) { \
- register type1 __arg1 asm("2") = arg1; \
- register long __svcreas asm("2"); \
- long __res; \
- __asm__ __volatile__ ( \
- " .if %1 < 256\n" \
- " svc %b1\n" \
- " .else\n" \
- " la %%r1,%1\n" \
- " svc 0\n" \
- " .endif" \
- : "=d" (__svcreas) \
- : "i" (__NR_##name), \
- "0" (__arg1) \
- : _svc_clobber ); \
- __res = __svcreas; \
- __syscall_return(type,__res); \
-}
-
-#define _syscall2(type,name,type1,arg1,type2,arg2) \
-type name(type1 arg1, type2 arg2) { \
- register type1 __arg1 asm("2") = arg1; \
- register type2 __arg2 asm("3") = arg2; \
- register long __svcreas asm("2"); \
- long __res; \
- __asm__ __volatile__ ( \
- " .if %1 < 256\n" \
- " svc %b1\n" \
- " .else\n" \
- " la %%r1,%1\n" \
- " svc 0\n" \
- " .endif" \

```

[PATCH 6/7] remove all remaining \_syscallX macros

```
- : "=d" (__svcrs) \  
- : "i" (__NR_###name), \  
- "0" (__arg1), \  
- "d" (__-#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)  
-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  
- */
```

[PATCH 6/7] remove all remaining \_syscallX macros

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

```
=====  
--- linux-cg.orig/include/asm-sh/unistd.h 2006-08-27 21:36:06.000000000 +0200  
+++ linux-cg/include/asm-sh/unistd.h 2006-08-27 21:36:52.000000000 +0200  
@@ -306,122 +306,6 @@
```

```
#ifdef __KERNEL__
```

```
/* user-visible error numbers are in the range -1 - -124: see <asm-sh/errno.h> */
```

```
-  
-#define __syscall_return(type, res) \  
-do { \  
- if ((unsigned long)(res) >= (unsigned long)(-124)) { \  
- /* Avoid using "res" which is declared to be in register r0; \  
- errno might expand to a function call and clobber it. */ \  
- int __err = -(res); \  
- errno = __err; \  
- res = -1; \  
- } \  
- return (type) (res); \  
-} while (0)
```

```
/* XXX - _foo needs to be __foo, while __NR_bar could be _NR_bar. */
```

```
-#define _syscall0(type,name) \  
-type name(void) \  
-{ \  
-register long __sc0 __asm__ ("r3") = __NR_##name; \  
- __asm__ __volatile__ ("trapa #0x10" \  
- : "=z" (__sc0) \  
- : "0" (__sc0) \  
- : "memory"); \  
-__syscall_return(type,__sc0); \  
-}
```

```
-#define _syscall1(type,name,type1,arg1) \  
-type name(type1 arg1) \  
-{ \  
-register long __sc0 __asm__ ("r3") = __NR_##name; \  
-register long __sc4 __asm__ ("r4") = (long) arg1; \  
- __asm__ __volatile__ ("trapa #0x11" \  
- : "=z" (__sc0) \  
- : "0" (__sc0), "r" (__sc4) \  
- : "memory"); \  
-__syscall_return(type,__sc0); \  
-}
```

```
-#define _syscall2(type,name,type1,arg1,type2,arg2) \  
-type name(type1 arg1,type2 arg2) \  
-{ \  
-register long __sc0 __asm__ ("r3") = __NR_##name; \  
-register long __sc4 __asm__ ("r4") = (long) arg1; \  
-
```

[PATCH 6/7] remove all remaining \_syscallX macros

[PATCH 6/7] remove all remaining \_syscallX macros

```
-register long __sc5 __asm__ ("r5") = (long) arg2; \  
-__asm__ __volatile__ ("tra
```