


Comparing changes


Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).





base: c01c4be6337248b2cf10c5ebd... ▾


←




compare: master ▾

 2 commits



 9 files changed

 0 commit comments

 1 contributor


-  Commits on Feb 28, 2018
-   pzhzqt

Added syscall.

b3eda3f
-   pzhzqt

removed unimportant code

e00d3cf

 Showing 9 changed files with 41 additions and 0 deletions.

Unified

Split

2 ■■■■ Makefile

174	_usertests\	174	_usertests\
175	_wc\	175	_wc\
176	_zombie\	176	_zombie\
177		177	+ _test\
178	fs.img: mkfs README \$(UPROGS)	178	fs.img: mkfs README \$(UPROGS)
179	./mkfs fs.img README \$(UPROGS)	179	./mkfs fs.img README \$(UPROGS)
244	mkfs.c ulib.c user.h cat.c echo.c forktest.c grep.c	244	mkfs.c ulib.c user.h cat.c echo.c forktest.c grep.c
kill.c\		245	kill.c\
245	ln.c ls.c mkdir.c rm.c stressfs.c usertests.c wc.c	246	ln.c ls.c mkdir.c rm.c stressfs.c usertests.c wc.c
zombie.c\		247	zombie.c\
246	printf.c umalloc.c\	247	printf.c umalloc.c\
		248	+ test.c\
247	README dot-bochsrc *.pl toc.* runoff runoff1	249	README dot-bochsrc *.pl toc.* runoff runoff1
runoff.list\		250	runoff.list\
248	.gdbinit.tmpl gdbutil\	251	.gdbinit.tmpl gdbutil\
249			

5 ■■■■ proc.c

112	memset(p->context, 0, sizeof *p->context);	112	memset(p->context, 0, sizeof *p->context);
113	p->context->eip = (uint)forkret;	113	p->context->eip = (uint)forkret;
114		114	
		115	+ int i;
		116	+ for(i=1;i<23;i++){
		117	+ p->cnt[i]=0;
		118	+ }
		119	+
115	return p;	120	return p;
116	}	121	}
117		122	

2 ■■■■ proc.h

...	@@ -1,3 +1,4 @@	1	+#include "syscall.h"
1	// Per-CPU state	2	// Per-CPU state
2	struct cpu {	3	struct cpu {
3	uchar apicid; // Local APIC ID	4	uchar apicid; // Local APIC ID

49	<code>struct file *ofile[NOFILE]; // Open files</code>	50	<code>struct file *ofile[NOFILE]; // Open files</code>
50	<code>struct inode *cwd; // Current directory</code>	51	<code>struct inode *cwd; // Current directory</code>
51	<code>char name[16]; // Process name (debugging)</code>	52	<code>char name[16]; // Process name (debugging)</code>
		53	<code>+ int cnt[23]; // count table of system</code>
			<code>calls</code>
52	<code>};</code>	54	<code>};</code>
53		55	
54	<code>// Process memory is laid out contiguously, low addresses first:</code>	56	<code>// Process memory is laid out contiguously, low addresses first:</code>

3 ■■■■ syscall.c			
103	<code>extern int sys_wait(void);</code>	103	<code>extern int sys_wait(void);</code>
104	<code>extern int sys_write(void);</code>	104	<code>extern int sys_write(void);</code>
105	<code>extern int sys_uptime(void);</code>	105	<code>extern int sys_uptime(void);</code>
		106	<code>+extern int sys_count(void);</code>
106		107	
107	<code>static int (*syscalls[])(void) = {</code>	108	<code>static int (*syscalls[])(void) = {</code>
108	<code>[SYS_fork] sys_fork,</code>	109	<code>[SYS_fork] sys_fork,</code>
126	<code>[SYS_link] sys_link,</code>	127	<code>[SYS_link] sys_link,</code>
127	<code>[SYS_mkdir] sys_mkdir,</code>	128	<code>[SYS_mkdir] sys_mkdir,</code>
128	<code>[SYS_close] sys_close,</code>	129	<code>[SYS_close] sys_close,</code>
		130	<code>+ [SYS_getCallCount] sys_count,</code>
129	<code>};</code>	131	<code>};</code>
130		132	
131	<code>void</code>	133	<code>void</code>
136		138	
137	<code>num = curproc->tf->eax;</code>	139	<code>num = curproc->tf->eax;</code>
138	<code>if(num > 0 && num < NELEM(syscalls) && syscalls[num]) {</code>	140	<code>if(num > 0 && num < NELEM(syscalls) && syscalls[num]) {</code>
		141	<code>+ curproc->cnt[num]++;</code>
139	<code>curproc->tf->eax = syscalls[num]();</code>	142	<code>curproc->tf->eax = syscalls[num]();</code>
140	<code>} else {</code>	143	<code>} else {</code>
141	<code>cprintf("%d %s: unknown sys call %d\n",</code>	144	<code>cprintf("%d %s: unknown sys call %d\n",</code>

1 ■■■■ syscall.h			
20	<code>#define SYS_link 19</code>	20	<code>#define SYS_link 19</code>
21	<code>#define SYS_mkdir 20</code>	21	<code>#define SYS_mkdir 20</code>
22	<code>#define SYS_close 21</code>	22	<code>#define SYS_close 21</code>
		23	<code>+ #define SYS_getCallCount 22</code>

13 ■■■■ sysproc.c			
89	<code>release(&tickslock);</code>	89	<code>release(&tickslock);</code>
90	<code>return xticks;</code>	90	<code>return xticks;</code>
91	<code>}</code>	91	<code>}</code>
		92	<code>+ </code>
		93	<code>+int</code>
		94	<code>+sys_count(void)</code>
		95	<code>+{</code>
		96	<code>+ int n;</code>
		97	<code>+ if(argint(0,&n)<0){</code>
		98	<code>+ return -1;</code>
		99	<code>+ }else if(n<1 n>22){</code>
		100	<code>+ return -1;</code>
		101	<code>+ }else{</code>
		102	<code>+ return myproc()->cnt[n];</code>
		103	<code>+ }</code>
		104	<code>+}</code>

13 ■■■■ test.c			
----------------	--	--	--

...	@@ -0,0 +1,13 @@	
		<pre>1 + #include "types.h" 2 + #include "user.h" 3 + #include "fcntl.h" 4 + #include "syscall.h" 5 + 6 + int main(void){ 7 + if (fork()==0){ 8 + printf(1,"fork called %d times in child process\n",getCallCount(SYS_fork)); 9 + }else{ 10 + printf(1,"fork called %d times in parent process\n",getCallCount(SYS_fork)); 11 + } 12 + exit(); 13 + }</pre>

1 ■■■■ user.h		
23	char* sbrk(int);	23 char* sbrk(int);
24	int sleep(int);	24 int sleep(int);
25	int uptime(void);	25 int uptime(void);
26		26 +int getCallCount(int);
27	// ulib.c	27
28	int stat(char*, struct stat*);	28 // ulib.c
		29 int stat(char*, struct stat*);

1 ■■■■ usys.S		
29	SYSCALL(sbrk)	29 SYSCALL(sbrk)
30	SYSCALL(sleep)	30 SYSCALL(sleep)
31	SYSCALL(uptime)	31 SYSCALL(uptime)
		32 +SYSCALL(getCallCount)

No commit comments for this range