Guide – Lab06

Es 2

In order to add system calls, a lot of files need to be modified:

param.h

Add anywhere this define (can be used in file.c to create a static array of conditions):

```
#define NCOND_MAX    20  // max number of cond
```

user.h

Add anywhere (in order to be able to call also those methods in a user program):

```
int cond_alloc();
void cond_set(int cond, int val);
int cond_get(int cond);
void cond_destroy(int cond);
void cond_wait(int cond);
void cond_signal(int cond);
void cond_broadcast(int cond);
```

usys.S

Add anywhere:

```
SYSCALL(cond_alloc)
SYSCALL(cond_set)
SYSCALL(cond_get)
SYSCALL(cond_destroy)
SYSCALL(cond_wait)
SYSCALL(cond_signal)
SYSCALL(cond_broadcast)
```

syscall.h

Add anywhere these to define the number associated with the newly created system calls. This will be the values that the num variable will take into the function syscall (see exercise 1). It is important that the numbers are not associated to others system calls:

```
#define SYS_cond_alloc     27
#define SYS_cond_set       28
#define SYS_cond_get       29
#define SYS_cond_destroy   30
#define SYS_cond_wait      31
#define SYS_cond_signal    32
#define SYS_cond_broadcast 33
```

syscall.c

Add this extern prototypes where a lot of other similar lines are written (line 100+something):

```
extern int sys_cond_alloc(void);
extern int sys_cond_set(void);
extern int sys_cond_get(void);
extern int sys_cond_destroy(void);
extern int sys_cond_wait(void);
extern int sys_cond_signal(void);
extern int sys_cond_broadcast(void);
```

And inside `static int (*syscalls[])(void) = { … }` (it is the initialization of an array with the pointers to the routines to be called) add those lines:

```
[SYS_cond_alloc]     sys_cond_alloc,
[SYS_cond_set]       sys_cond_set,
[SYS_cond_get]       sys_cond_get,
[SYS_cond_destroy]   sys_cond_destroy,
[SYS_cond_wait]      sys_cond_wait,
[SYS_cond_signal]    sys_cond_signal,
[SYS_cond_broadcast] sys_cond_broadcast
```

main.c

add after `semaphore_init();` (line 35) this line to initialize the management of the condition (this function must be generated in the file.c, see below):

```
condition_init();
```

defs.h

add after `void semaphore_init(void);` in order to be able to call this function also from the main.c:

```
void condition_init(void);
```

cond_test.c (new file)

This file does not exist yet, you need to create it.But for now simply run **cp st.c cond_test.c** since this file will contain a test program that will use the condition system calls.

Makefile

Add after `_st\` (line 167) the following, in order to add the cond_test program to the list of user programs that will be available inside qemu:

```
_\cond_test
```

sysfile.c

Inside this file you need to add some functions that act as wrappers of some other functions (the ones that really are responsible for managing the conditions, inside file.c). The wrappers have a fixed signature: int f(void) and inside them there should be the retreival of parameters (using the function argint), the calling of the corresponding function belonging to the file file.c, and the translation of the return value. You can write them for exaple after line 88:

```
int sys_cond_alloc(void) {
   int cn;
  cn = cond_alloc();
  return cn;
}
int sys_cond_set(void) {
   int cn, n;
```

```
    if(argint(0, &cn) < 0 || argint(1, &n) < 0) {
        return -1;
    }
    cond_set(cn, n);
    return 0;
}
int sys_cond_get(void) {
    int cn, n;
    if(argint(0, &cn) < 0) {
        return -1;
    }
    n = cond_get(cn);
    return n;
}
int sys_cond_destroy(void) {
    int cn;
    if(argint(0, &cn) < 0) {
        return -1;
    }
    cond_destroy(cn);
    return 0;
}
int sys_cond_wait(void) {
    int cn;
    if(argint(0, &cn) < 0) {
        return -1;
    }
    cond_wait(cn);
    return 0;
}
int sys_cond_signal(void) {
    int cn;
    if(argint(0, &cn) < 0) {
        return -1;
    }
    cond_signal(cn);
    return 0;
}
int sys_cond_broadcast(void) {
    int cn;
    if(argint(0, &cn) < 0) {
        return -1;
    }
    cond_broadcast(cn);
    return 0;
}
```

file.c

This is the most important file, that will contain the implementation of those functions.

In this file we add a struct condition declaration:

```
struct condition {
   // TODO add something there
   struct spinlock lock;
};
```

A global condition table, that will contain an array of struct condition:

```
struct {
   struct spinlock lock;
   struct condition condition[NCOND_MAX];
} condition_table;
```

The function called by the main to initialize the condition_table:

```
void condition_init(void)
{
  initlock(&condition_table.lock, "condition_table");
}
```

And all the functions:

```
int cond_alloc(void) {
  // TODO
  return 0;
}
void cond_set(int cn, int n) {
// TODO
}
int cond_get(int cn) {
  // TODO
  return 0;
}
void cond_destroy(int cn) {
  // TODO
}
void cond_wait(int cn) {
   // TODO
}
void cond_signal(int cn) {
  // TODO
}
void cond_broadcast(int cn) {
  // TODO
}
```