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  
}