Donghyun Park

Prof. Hakner

Operating Systems

12/27/17

PSet 6 – Synchronization

Problem 1 – Spin Lock

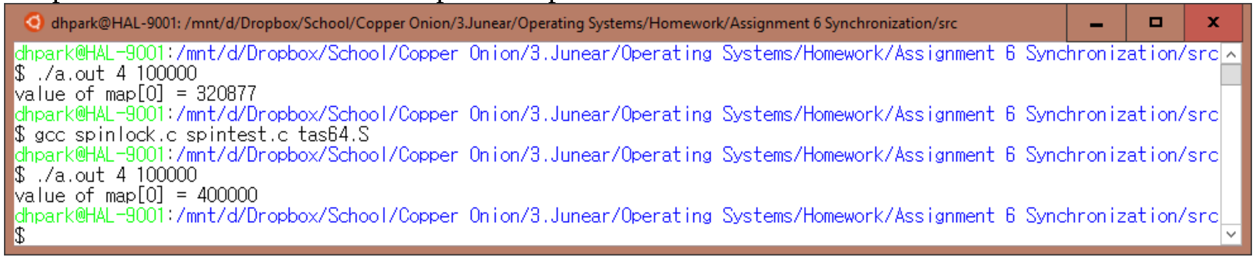
spinlock.h:
```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <signal.h>
#include <sys/types.h>
#include <sys/mman.h>
#include <sys/stat.h>
#include <sys/wait.h>

typedef struct spinlock{
  volatile char primlock;
}spinlock;

int tas(volatile char *lock);

void spin_lock(struct spinlock *l);
void spin_unlock(struct spinlock *l);
```

spinlock.c:
```
#include "spinlock.h"

void spin_lock(struct spinlock *l){
  while(tas(&(l->primlock)) != 0){
    ;
  }
}

void spin_unlock(struct spinlock *l){
  l->primlock = 0;
}
```

Problem 2 – Test the test-and-set

spintest.c:
```
#include "spinlock.h"

int main(int argc, char **argv){
  long long unsigned int nchild, niter;
  pid_t pid;

  if(argc != 3){
    fprintf(stderr, "Error: Please input 3 arguments\n");
    exit(255);
  }

  nchild = atoll(argv[1]);
  niter = atoll(argv[2]);

  int *map = mmap(NULL, 4096, PROT_READ|PROT_WRITE,
MAP_ANONYMOUS|MAP_SHARED, 0, 0);
  if(map < 0){
    fprintf(stderr, "Error: failed to mmap() anonymous page
- %s\n", strerror(errno));
    exit(255);
  }

  map[0] = 0;
  spinlock *lock;
  lock = (spinlock *)(map + sizeof(spinlock));
  lock -> primlock = map[1];

  for(int i = 0; i < nchild; i++){
    switch(pid = fork()){
      case -1:
        fprintf(stderr, "Error: failed to fork() #%dth time
- %s\n", i, strerror(errno));
        break;
      case 0:
        spin_lock(lock);
        for(int j = 0; j < niter; j++){
          map[0]++;
        }
        spin_unlock(lock);
        exit(0);
        break;
    }
  }
  for(int i = 0; i < nchild; i++){
    wait(0);
  }

  printf("value of map[0] = %d\n", map[0]);
  return 0;
}
```

Output of without and then with spin lock protection:



```
dhpark@HAL-9001:/mnt/d/Dropbox/School/Copper Onion/3.Junear/Operating Systems/Homework/Assignment 6 Synchronization/src
$ ./a.out 4 100000
value of map[0] = 320877
dhpark@HAL-9001:/mnt/d/Dropbox/School/Copper Onion/3.Junear/Operating Systems/Homework/Assignment 6 Synchronization/src
$ gcc spinlock.c spintest.c tas64.S
dhpark@HAL-9001:/mnt/d/Dropbox/School/Copper Onion/3.Junear/Operating Systems/Homework/Assignment 6 Synchronization/src
$ ./a.out 4 100000
value of map[0] = 400000
dhpark@HAL-9001:/mnt/d/Dropbox/School/Copper Onion/3.Junear/Operating Systems/Homework/Assignment 6 Synchronization/src
$
```

<u>Problem 3 – Implement Condition Variables</u>

<u>cv.h:</u>

```
#include "spinlock.h"

typedef struct cv{
  int i;
  spinlock lock;
  pid_t pid[64];
  sigset_t sigMask;
}cv;

void cv_init(struct cv *cv);
void cv_wait(struct cv *cv, struct spinlock *mutex);
int cv_broadcast(struct cv *cv);
int cv_signal(struct cv *cv);
```

<u>cv.c:</u>

```
#include "cv.h"

void sigHandle(int signum){
  ;
}

void cv_init(struct cv *cv){
  spinlock *lock;

  int *map = mmap(NULL, 4096, PROT_READ|PROT_WRITE,
MAP_ANONYMOUS|MAP_SHARED, 0, 0);
```

```
  if(map < 0){
    fprintf(stderr, "Error: failed to mmap() anonymous page
- %s\n", strerror(errno));
    exit(255);
  }

  lock = (spinlock *)(map + sizeof(spinlock));
  cv->lock = *lock;

  for(int i = 0; i < 64; i++){
    cv->pid[i] = 0;
  }

  cv->i = 0;
  signal(SIGUSR1, sigHandle);
  sigfillset(&cv->sigMask);
  sigdelset(&cv->sigMask, SIGUSR1);
}

void cv_wait(struct cv *cv, struct spinlock *mutex){
  if(cv->i >= 64){
     fprintf(stderr, "Error: too many processes\n");
     exit(255);
  }

  spin_lock(&cv->lock);
  cv->pid[cv->i] = getpid();
  cv->i++;
  spin_unlock(&cv->lock);
  spin_unlock(mutex);

  sigprocmask(SIG_BLOCK, &cv->sigMask, NULL);
  sigsuspend(&cv->sigMask);

  if(cv->i > 0){
     spin_lock(&cv->lock);
     cv->pid[cv->i - 1] = 0;
     cv->i--;
     spin_unlock(&cv->lock);
     spin_lock(mutex);
  return;
  }

  sigprocmask(SIG_UNBLOCK, &cv->sigMask, NULL);
  spin_lock(mutex);
}
```

```
int cv_broadcast(struct cv *cv){
  int wakeNum = 0;

  spin_lock(&cv->lock);

  if(cv->i == 0){
    spin_unlock(&cv->lock);
    return 0;
  }

  for(int j = 0; j < 64; j++){
    if(cv->pid[j] > 0){
      kill(cv->pid[j], SIGUSR1);
      wakeNum++;
    }
  }
  spin_unlock(&cv->lock);
  return wakeNum;
}

int cv_signal(struct cv *cv){
  int wakeNum = 0;

  spin_lock(&cv->lock);

  if(cv->i == 0){
    spin_unlock(&cv->lock);
    return 0;
  }

  kill(cv->pid[cv->i - 1], SIGUSR1);
  wakeNum++;
  spin_unlock(&cv->lock);
  return wakeNum;
}
```

Problem 4 – A FIFO using condition variables

fifo.h:

```
#include "cv.h"

#define MYFIFO_BUFSIZ 1024

typedef struct fifo{
  int state, readNext, writeNext;
  unsigned long buf[MYFIFO_BUFSIZ];
  spinlock lock;
  cv w, r;
}fifo;

void fifo_init(struct fifo *f);
void fifo_wr(struct fifo *f, unsigned long d);
unsigned long fifo_rd(struct fifo *f);
```

<u>fifo.c:</u>

```
#include "fifo.h"

void fifo_init(struct fifo *f){
  cv *readMap = NULL, *writeMap = NULL;

  readMap = (cv *)mmap(NULL, sizeof(cv), PROT_READ|PROT_WRITE,
MAP_SHARED|MAP_ANONYMOUS, -1, 0);
  writeMap = (cv *)mmap(NULL, sizeof(cv), PROT_READ|PROT_WRITE,
MAP_SHARED|MAP_ANONYMOUS, -1, 0);
  if(readMap < 0){
       fprintf(stderr, "Error: failed to mmap() anonymous file
for read %s\n", strerror(errno));
       exit(255);
  }
  if(writeMap < 0){
    fprintf(stderr, "Error: failed to mmap() anonymous file for
write %s\n", strerror(errno));
    exit(255);
  }

  fifo->r = *readMap;
  fifo->readNext = 0;
  cv_init(&fifo->r);

  fifo->w = *writeMap;
  fifo->writeNext = 0;
  cv_init(&fifo->w);

  fifo->state = 0;
  fifo->lock.primlock = 0;
}

void fifo_wr(struct fifo *f, unsigned long d){
  spin_lock(&fifo->lock);

  while(fifo->state >= MYFIFO_BUFSIZ){
       cv_wait(&fifo->w, &fifo->lock);
  }
  fifo->buf[fifo->writeNext++] = x;
  fifo->writeNext %= MYFIFO_BUFSIZ;
  fifo->state++;

  cv_signal(&fifo->r);
  spin_unlock(&fifo->lock);
}

unsigned long fifo_rd(struct fifo *f){
  unsigned long fifoRead;

  spin_lock(&fifo->lock);

  while(fifo->state <= 0){
       printf("Fifo read is %d complete\n", ++z);
       cv_wait(&fifo->r, &fifo->lock);
  }

  fifoRead = fifo->buf[fifo->readNext++];
  fifo->readNext %= MYFIFO_BUFSIZ;
  fifo->state--;

  cv_signal(&fifo->w);
  spin_unlock(&fifo->lock);
  return fifoRead;
}
```

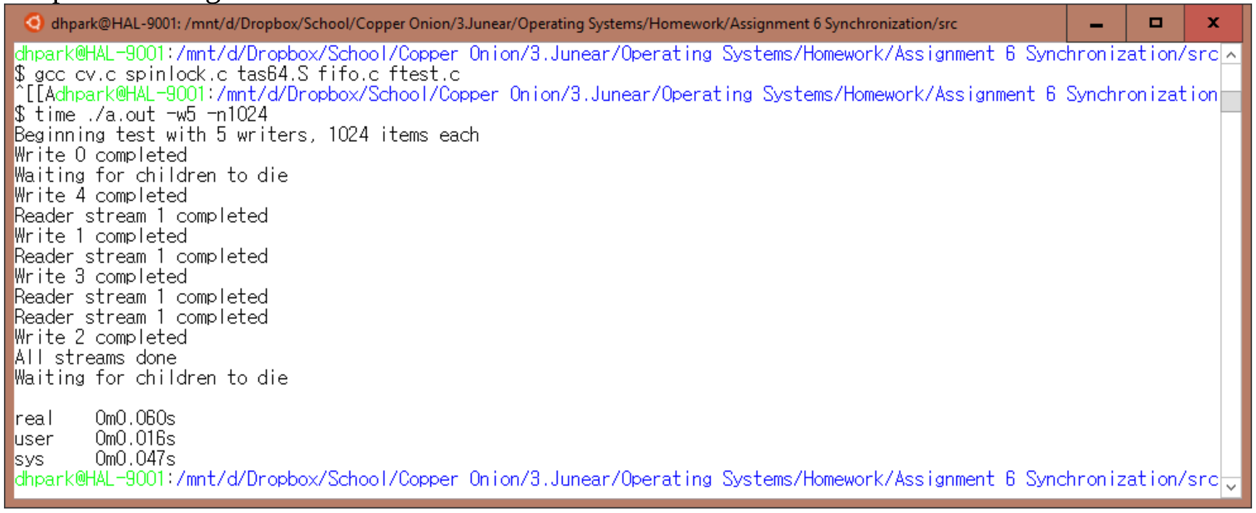
Problem 5 – Test your FIFO

ftest.c:

```
#include "fifo.h"

int main(int argc, char **argv){
  int writers, items;
  fifo *fifo1;
  pid_t pid1, pid2;

  if(argc < 3){
    printf("Error: please input 3 arguments\n");
    exit(255);
  }

  writers = atoi(argv[1]+2);
  items = atoi(argv[2]+2);
  printf("Beginning test with %d writers, %d items each\n",
writers, items);
```

```
  fifo1 = (fifo *)mmap(NULL, sizeof(fifo), PROT_READ|PROT_WRITE,
MAP_SHARED|MAP_ANONYMOUS, -1, 0);
  if(fifo1 < 0){
    fprintf(stderr, "Error: failed to mmap() anonymous page for
fifo - %s\n", strerror(errno));
    exit(255);
  }

  fifo_init(fifo1);

  for(int i = 0; i < writers; i++){
    switch(pid1 = fork()){
      case -1:
        fprintf(stderr, "Error: failed to fork() #%dth time
- %s\n", i, strerror(errno));
        break;
      case 0:;
        unsigned long writeBuf[items];

        for(int j = 0; j < items; j++){
          writeBuf[j] = getpid()*10000 + j;
          fifo_wr(fifo1, writeBuf[j]);
        }

        printf ("Write %d completed\n", i);
        exit(0);
        break;
    }
  }

  switch(pid2 = fork()){
    case -1:
      fprintf(stderr, "Error: failed to fork() - %s\n",
strerror(errno));
      break;
    case 0:;
      unsigned long readBuf[writers * items];

      for(int i = 0; i < (writers * items); i++){
        readBuf[i] = fifo_rd(fifo1);
      }

      printf("All streams done\n");
      break;
  }

  printf("Waiting for writer children to die\n");
  for(int i = 0; i < writers + 1; i++){
    wait(0);
  }

  return 0;
}
```

Output of testing the FIFO:



dhpark@HAL-9001: /mnt/d/Dropbox/School/Copper Onion/3.Junear/Operating Systems/Homework/Assignment 6 Synchronization/src

```
dhpark@HAL-9001:/mnt/d/Dropbox/School/Copper Onion/3.Junear/Operating Systems/Homework/Assignment 6 Synchronization/src
$ gcc cv.c spinlock.c tas64.S fifo.c ftest.c
^[[Adhpark@HAL-9001:/mnt/d/Dropbox/School/Copper Onion/3.Junear/Operating Systems/Homework/Assignment 6 Synchronization
$ time ./a.out -w5 -n1024
Beginning test with 5 writers, 1024 items each
Write 0 completed
Waiting for children to die
Write 4 completed
Reader stream 1 completed
Write 1 completed
Reader stream 1 completed
Write 3 completed
Reader stream 1 completed
Reader stream 1 completed
Write 2 completed
All streams done
Waiting for children to die

real    0m0.060s
user    0m0.016s
sys     0m0.047s
dhpark@HAL-9001:/mnt/d/Dropbox/School/Copper Onion/3.Junear/Operating Systems/Homework/Assignment 6 Synchronization/src
```