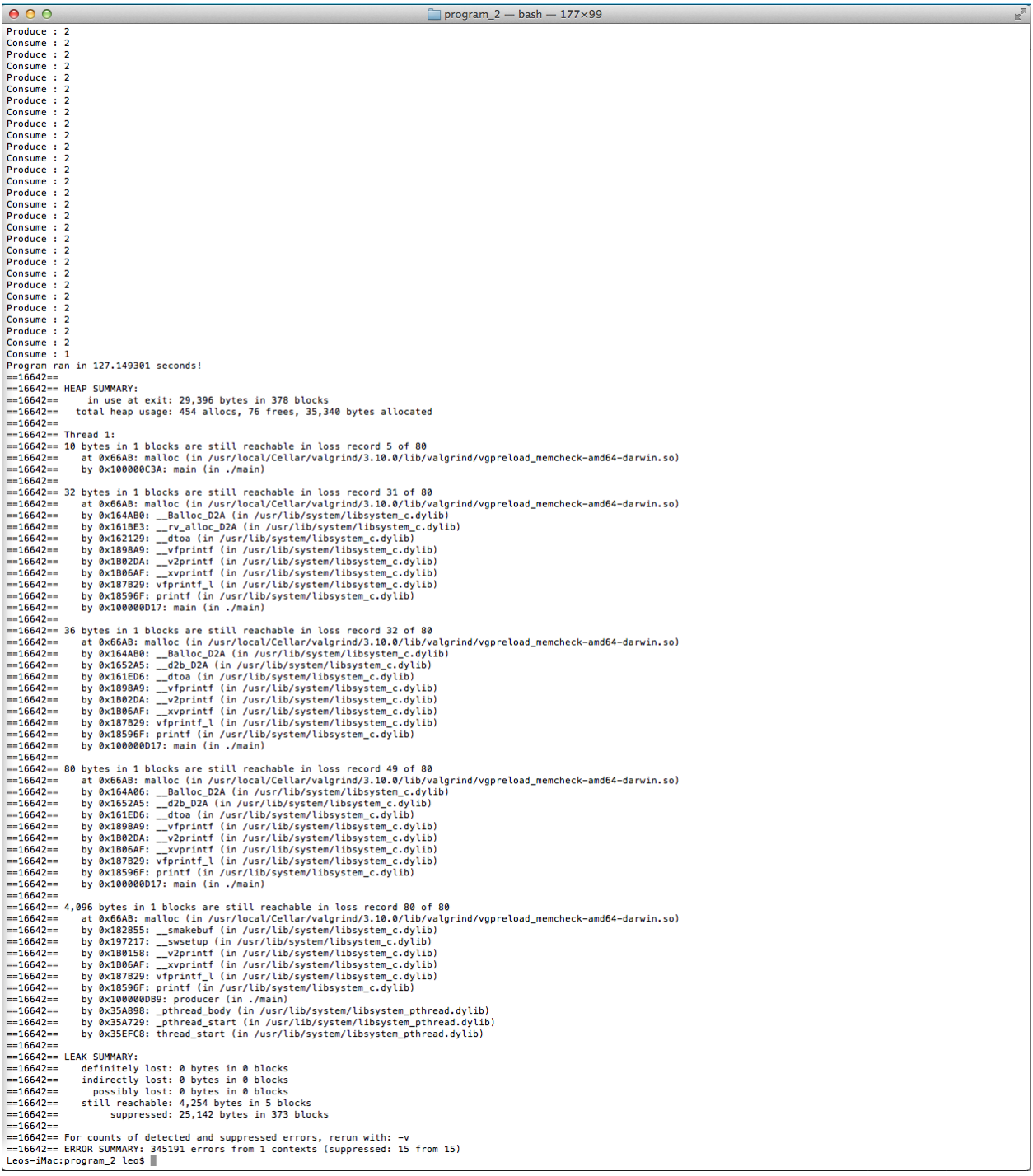
Leo Schultz
10/8/14
Parallel Programming HW #2
Consumer-Producer Problem with Valgrind



```
// C Implementation

# include <stdio.h>
# include <stdlib.h>
# include <pthread.h>
# include <sys/time.h>

# define BufferSize 10
# define MAX 2000000

void *producer();
void *consumer();

int BufferIndex=0;
char *BUFFER;

pthread_cond_t Buffer_Not_Full=PTHREAD_COND_INITIALIZER;
pthread_cond_t Buffer_Not_Empty=PTHREAD_COND_INITIALIZER;
pthread_mutex_t mVar=PTHREAD_MUTEX_INITIALIZER;

int main()
{
   struct timeval tim;
   gettimeofday(&tim, NULL);
   double start_time=tim.tv_sec+(tim.tv_usec/1000000.0);

   pthread_t ptid,ctid;
   BUFFER=(char *) malloc(sizeof(char) * BufferSize);
   pthread_create(&ptid,NULL,producer,NULL);
   pthread_create(&ctid,NULL,consumer,NULL);
   pthread_join(ptid,NULL);
   pthread_join(ctid,NULL);

   gettimeofday(&tim, NULL);
   double end_time=tim.tv_sec+(tim.tv_usec/1000000.0);
   double runtime = end_time - start_time;
   printf("Program ran in %f seconds! \n", runtime);
   return 0;
}

void *producer()
{
   int i;
   for(i = 1; i <= MAX; i++)
   {
      pthread_mutex_lock(&mVar);
      if(BufferIndex==BufferSize)
      {
         pthread_cond_wait(&Buffer_Not_Full,&mVar);
```

```
        }
        BUFFER[BufferIndex++]='@';
        printf("Produce : %d \n",BufferIndex);
        pthread_mutex_unlock(&mVar);
        pthread_cond_signal(&Buffer_Not_Empty);
    }
    return 0;
}

void *consumer()
{
    int i;
    for(i = 1; i <= MAX; i++)
    {
        pthread_mutex_lock(&mVar);
        if(BufferIndex==-1)
        {
            pthread_cond_wait(&Buffer_Not_Empty,&mVar);
        }
        printf("Consume : %d \n",BufferIndex--);
        pthread_mutex_unlock(&mVar);
        pthread_cond_signal(&Buffer_Not_Full);
    }
    return 0;
}
```