# ELEC 377 – Operating Systems

Lab 3 Tutorial

# Lab 3

- Purpose 1 – implement synchronization
- Purpose 2 – shared memory on Unix
- Purpose 3 – separate compilation

# Lab 3

- Skeleton Code is in SVN and on the handouts page of the web site
  ◊ 6 files:
  -producer.c, consumer.c
  -common.h, common.c
  -makefile, meminit.c

- Makefile is used to simplify building systems
  *make* - with no args builds entire system
  *make xxx* - builds xxx

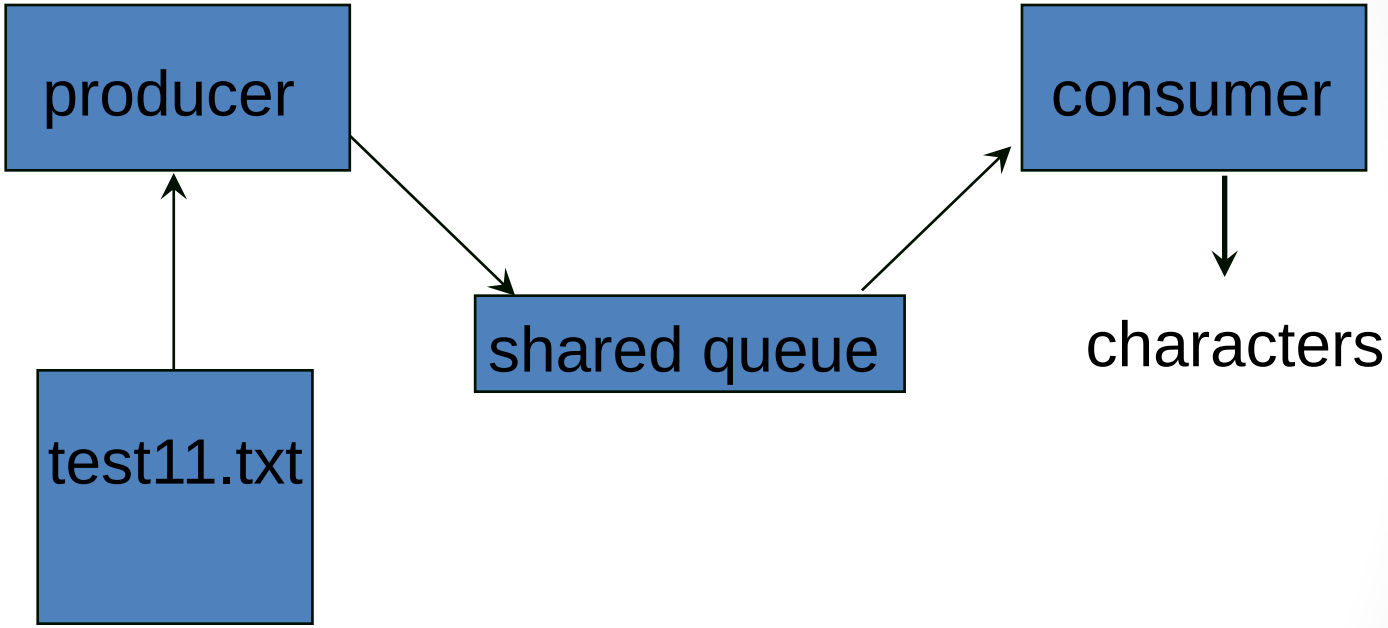
# Lab 3 - User Level Code

- Programs run at the user level
- Not Kernel dependent
- Can be done on any version of Unix with shared memory segments
  - Linux
  - Sun
  - Mac Os X

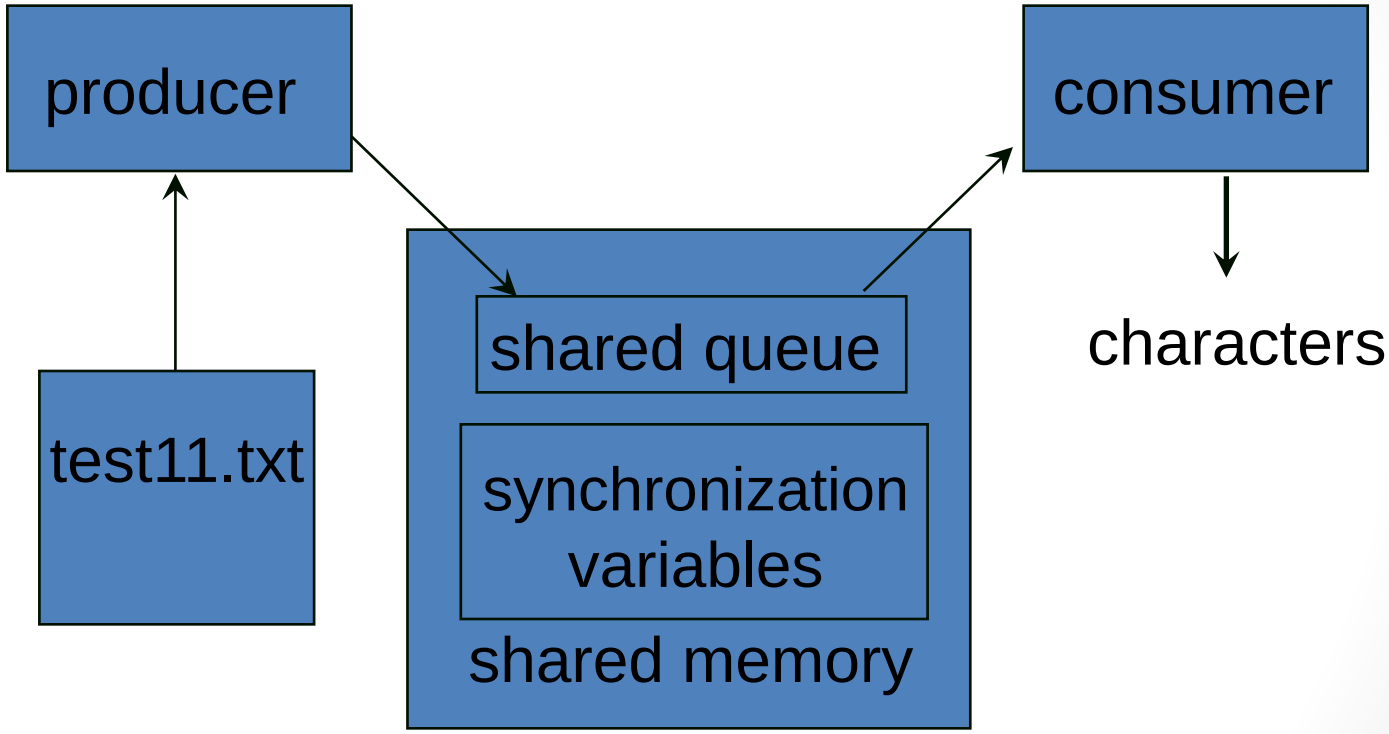
# Lab 3 - System Structure

- Producer, Consumer
- Producer reads a file of characters and passes it to the consumer through the shared memory
- Consumer reads characters through the shared memory and prints them out

# Lab 3 - System Structure



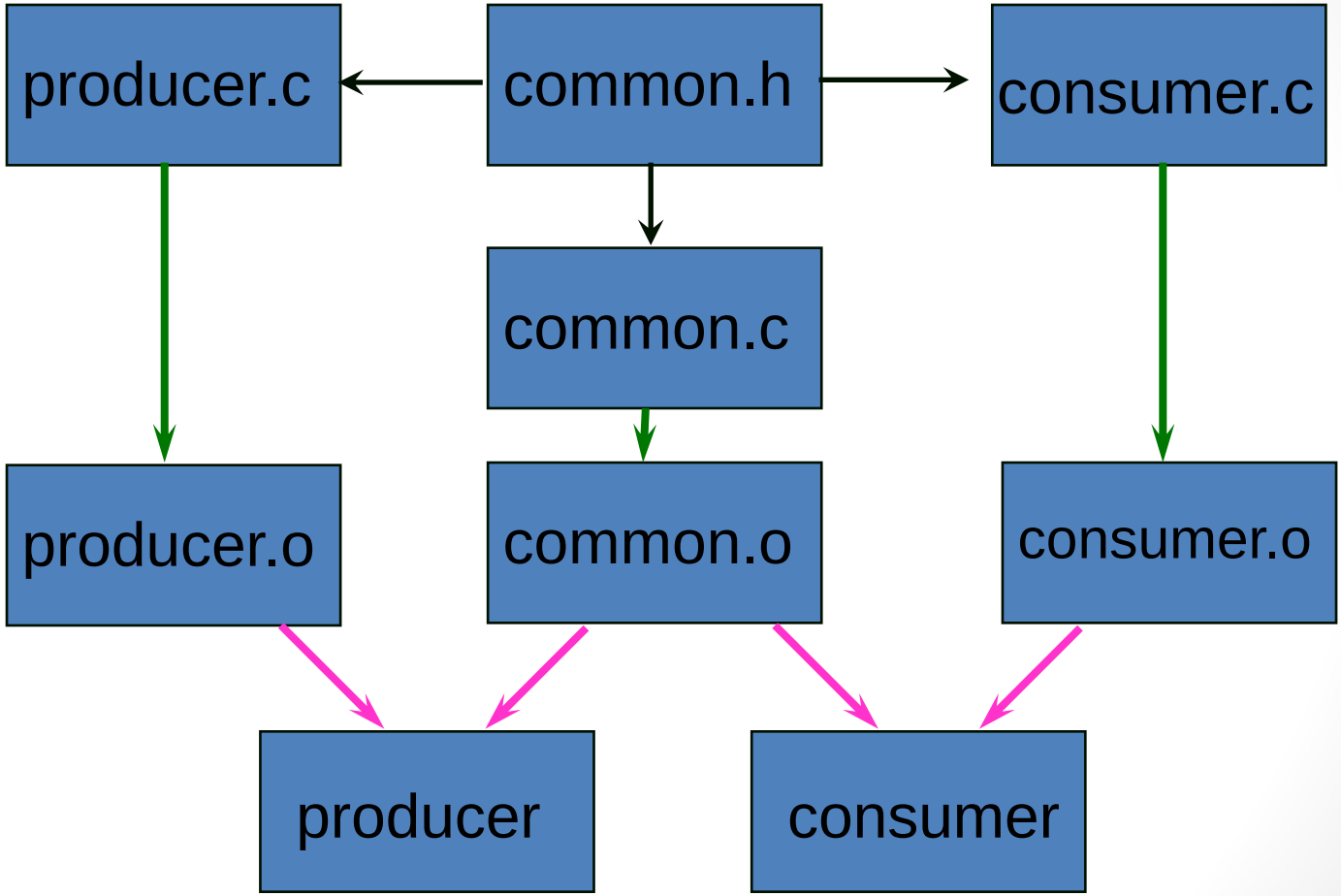
# Lab 3 - System Structure



# Lab 3 - Compile Time Structure

- mutexes are common code
◊ same code for both producer and consumer
◊ put in a common file so that both can use same routines
- also need to impose structure on the share memory
◊ shmat returns void *
  - pointer to arbitrary memory
◊ both consumer and producer have to have the same view of the shared memory
  - structure definition is common to producer and consumer

# Lab 3 - Compile Structure



# Lab 3 - Shared Memory Structure

```
struct shared {
  /* synchronization variables */
  int turn[2];
  int flag[2];
  int endOfFile;

  /* queue variables */
  char buffer[BUFFSIZE];
  int in;
  int out;
  int count;
}
```

# Lab 3 - Common.c

```
void getMutex(int pid){
}

void releaseMutex(int pid){
}
```

use the shared pointer to get at the shared sychronization control variables!!

*pid* is the same as *i* in the algorithm

# Lab 3 - When to stop

- Use a flag in the shared memory structure to indicate when the data is finished

- Producer reads until end of file, then sets the end of file marker.

- Consumer

◊ When the queue is empty, check the end of file marker.

# Lab 3 - What you have to do

- common.c
  - ◊ write getMutex and releaseMutex based on Bakery solution
- producer.c
  - ◊ read data from file
  - ◊ add data to queue
- consumer.c
  - ◊ read data from queue
  - ◊ write to std out
- Transfer data one item at a time
- Nested Loops in both producer and consumer
- ***All*** access to shared data is guarded by the mutex
- No I/O in the critical section!!

# Lab 3 -Testing

- Think about your data
  ◊ your data should prove that
    - all of the data is transferred
  - only the data is transferred
    - the order of the data is preserved
    - no duplicates are introduced.