Elikem Hermon | 20075527
Hermann Krohn | 20057435

Lab 3 – Description of Program

The program allows multiple producers and consumers to access a buffer that exists in shared memory. To ensure mutual exclusion within the shared memory segment, the pass the key algorithm is implemented using test and set with the aid of synchronization variables which also exist in the shared memory segment.

The program allocates the shared memory segment using the meminit sub-program. The shared memory segment is allocated using the shmget function. It is then mapped to the shared struct (memptr) using the shmat function. The entries of this struct are then initialized to zero using a for loop.

The synchronization logic for the program exists in the common sub-program which is accessible to both the producer and consumer. The common sub-program has a mutexInit function that initializes the pointer to the shared memory segment. A getMutex function sets the waiting flag of the calling process to true and invokes the test and set function within a while loop. This loop terminates when either the waiting flag of the process or the key is false. The waiting flag is then set to false if the process gets access to the critical section.

The releaseMutex function checks if any another process is waiting to enter the critical section and sets the waiting flag of that process to false. In the case that no other process is waiting, it resets the lock to false.

Critical sections within the producer and consumer are marked by calls to getMutex at the top of a critical section and releaseMutex at the end. The producer sub-program begins by creating a pointer to the shared memory segment and initializing it using mutexInit. The variable which keeps track of the number of producers is then incremented within a critical section. A loop is then used to read characters from standard input until EOF is reached. An inner loop uses a condition variable (stored) that loops until a character is placed onto the shared buffer. A critical section exists within the inner loop that places a character onto the buffer if there is space. In addition, the index of the producer within the buffer (in) is incremented and the count of items is incremented. After all the characters are placed, the number of producers is decremented.

The consumer sub-program begins by creating a pointer to the shared memory segment and initializing it using mutexInit. Within an infinite loop, the number of items in the buffer as well as the number of producers are checked. If both are zero, the program breaks out of the loop. If the former condition is not satisfied, an inner loop is used to wait until a character is retrieved from the buffer. A critical section exists within this inner loop that reads the current character from the buffer if it is not empty. In addition, it increments the index of the consumer within the buffer (out) and decrements the number of items in the buffer. If the buffer is empty and the number of producers is zero or a letter is read, the program breaks out of the inner loop. Outside the inner loop, the character is sent to standard output using the putchar function.