```
#define NPIPE 8
#define PSIZE 20

typedef struct pipe
{
    char buf[PSIZE];
    int head, tail;
    int data, room;
    int status;
}PIPE;


PIPE *kpipe;

int create_pipe()
{
  PIPE *p = &kpipe;
  p->head = 0;
  p->tail = 0;
  p->data = 0;
  p->room = PSIZE;

}


pipe_init()
{

}

int write_pipe(PIPE *p, char buf[], int n)
{
   int r = 0;

  if(n<=0)//if no data
      return 0;

  while(n)
  {
    printf("writer %d writing pipe\n", running->pid);
    while(p->room)//loop while there is still room in the pipe
    {
        p->buf[p->head++] = *buf;//write byte to pipe
        p->head %= PSIZE;
        buf++;
        p->data++;//increase data
        p->room--;//decrease room
        r++;
        n--;

        if(n==0)//stop when all the bytes have been read
        {
          kwakeup(&p->data);
          return r;
        }
    printf("writer %d sleep for room\n", running->pid);
    kwakeup(&p->data);//wakeup the process that has to read the data
    ksleep(&p->room);//sleep process that sent data
    }

  }

}
```

```
int read_pipe(PIPE *p, char buf[], int n)
{
  int r = 0;

  if(n<=0)//no data
      return 0;


  while(n)//while bytes to be read
  {
    printf("reader %d reading pipe\n", running->pid);
    r=0;
    while(p->data)//while data left
    {
        *buf = p->buf[p->tail++];
        p->tail %= PSIZE;//unsure of this syntax
        buf++;
        p->data--;
        p->room ++;
        r++;
        n--;

        if(n==0)//if done reading
          break;
    }


    if(r)//has data
    {
      kwakeup(&p->room);//wakeup the proc
      return r;
    }

  printf("reader %d sleep for data", running->pid);
  kwakeup(&p->room);//wakeup reader
  ksleep(&p->data);//sleep writer
  continue;
  }
}

int check_writers()
{
        int yes = 0;
        for(int i = 1; i < NPROC; i++)
        {
                PROC * p = proc + i;
                if(p->pipe_flag == 1)
                {
                        yes = 1;
                        break;
                }
        }
        return yes;
}

int check_readers()
{
        int yes = 0;
        for(int i = 1; i < NPROC; i++)
        {
                PROC * p = proc + i;
                if(p->pipe_flag == 0)
                {
                        yes = 1;
```

```
                        break;
                }
        }
        return yes;
}
```