```
kern_loader_reply_t kern_loader_reply =
    0,  /* argument to pass to function */
    0,  /* timeout for rpc return msg_send */
    0,  /* string function */
    0,  /* ping function */
    log_data /* log_data function */


void listen(port_name_t port)

    charmsg_buf[kern_loader_replyMaxRequestSize]
    msg_header_t *msg = (msg_header_t *)msg_buf
    kern_return_t r

    while (1)









kern_return_t log_data(void *arg, printf_data_t log_data, unsigned int log_data_count)

    kern_return_t r

    /* Print the string we were passed, with our prefix. */
    printf("log_data: %s", log_data)

    /* Deallocate the memory used for the string. */
    vm_deallocate(task_self(), (vm_address_t)log_data,
   log_data_count*sizeof(*log_data))

    /* Get another log message. */
    r = kern_loader_get_log(kl_port, server_com_port, reply_port)
    if (r != KERN_SUCCESS)



    return KERN_SUCCESS
```

```
kern_loader_reply_t kern_loader_reply =
    0,  /* argument to pass to function */
    0,  /* timeout for rpc return msg_send */
    0,  /* string function */
    ping,    /* ping function */
    0   /* log_data function */


void ping_thread(port_name_t port)

    charmsg_buf[kern_loader_replyMaxRequestSize]
    msg_header_t *msg = (msg_header_t *)msg_buf
    kern_return_t r

    /* message handling loop */
    while (TRUE)









    /* We get here only if msg_receive returned an error. */
    mach_error("ping_thread receive", r)
    exit(1)


/* This function is called after a kern_loader_ping. */
kern_return_t ping (void *arg, int id)

    exit(0)    /* Kill this process. */
```

```
kern_loader_reply_t kern_loader_reply =
    0,/* argument to pass to function */
    0,/* timeout for rpc return msg_send */
    print_string,    /* string function */
    0,/* reply_ping function */
    0 /* log_data function */


void receive_thread(port_name_t port)

    charmsg_buf[kern_loader_replyMaxRequestSize]
    msg_header_t *msg = (msg_header_t *)msg_buf
    kern_return_t r

    /* message handling loop */
    while (TRUE)









    /* We get here only if msg_receive returned an error. */
    mach_error("receive_thread receive", r)
    exit(1)


/* Called every time kern_loader has status to report. */
kern_return_t print_string(void *arg, printf_data_t string,
    u_int string_count, int level)

    /* If the string is empty, return. */
    if (string_count == 0 || !string)
   return KERN_SUCCESS

    /* Print the string we were passed, with our prefix. */
    printf("print_string: %s", string)

    return KERN_SUCCESS
```