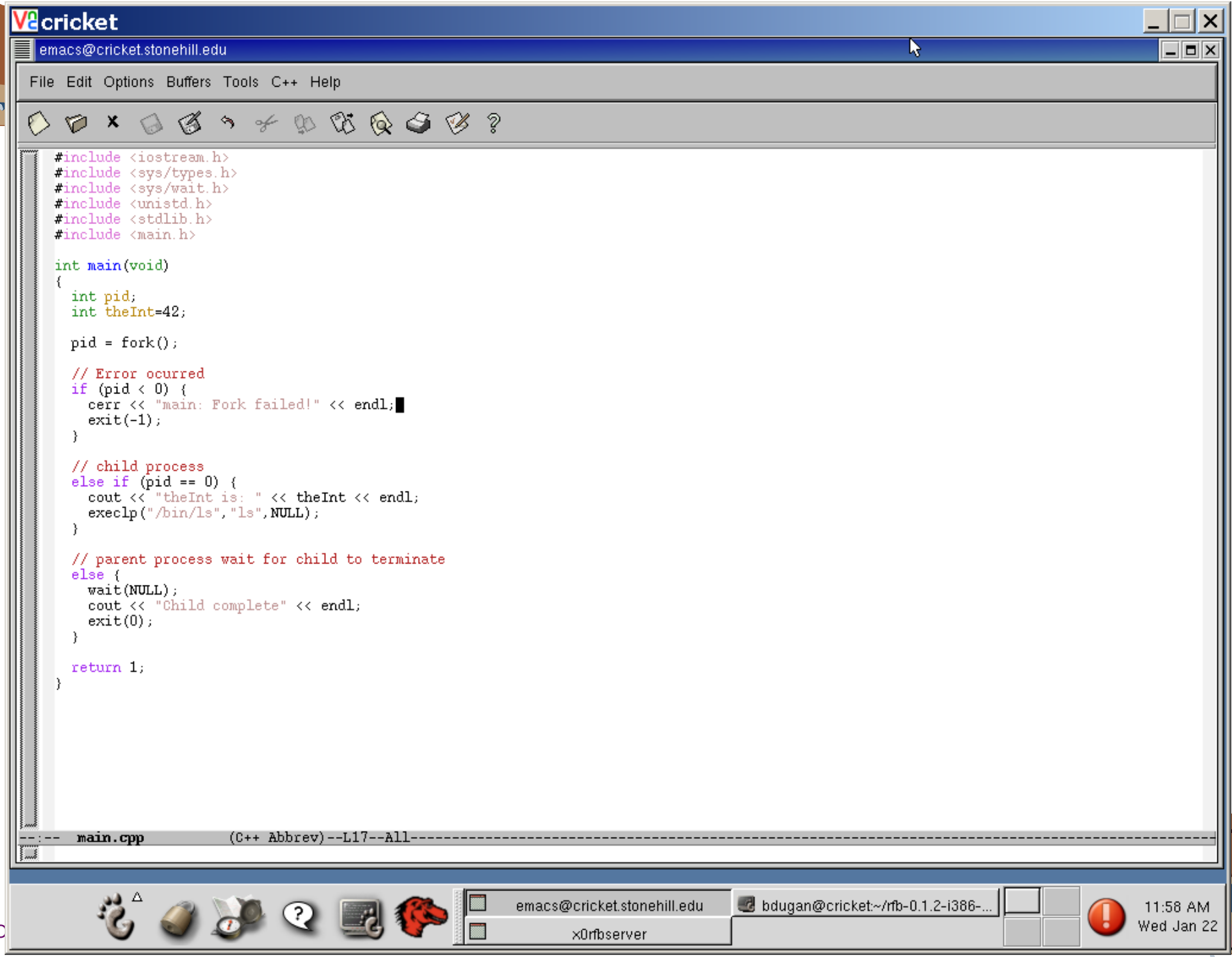
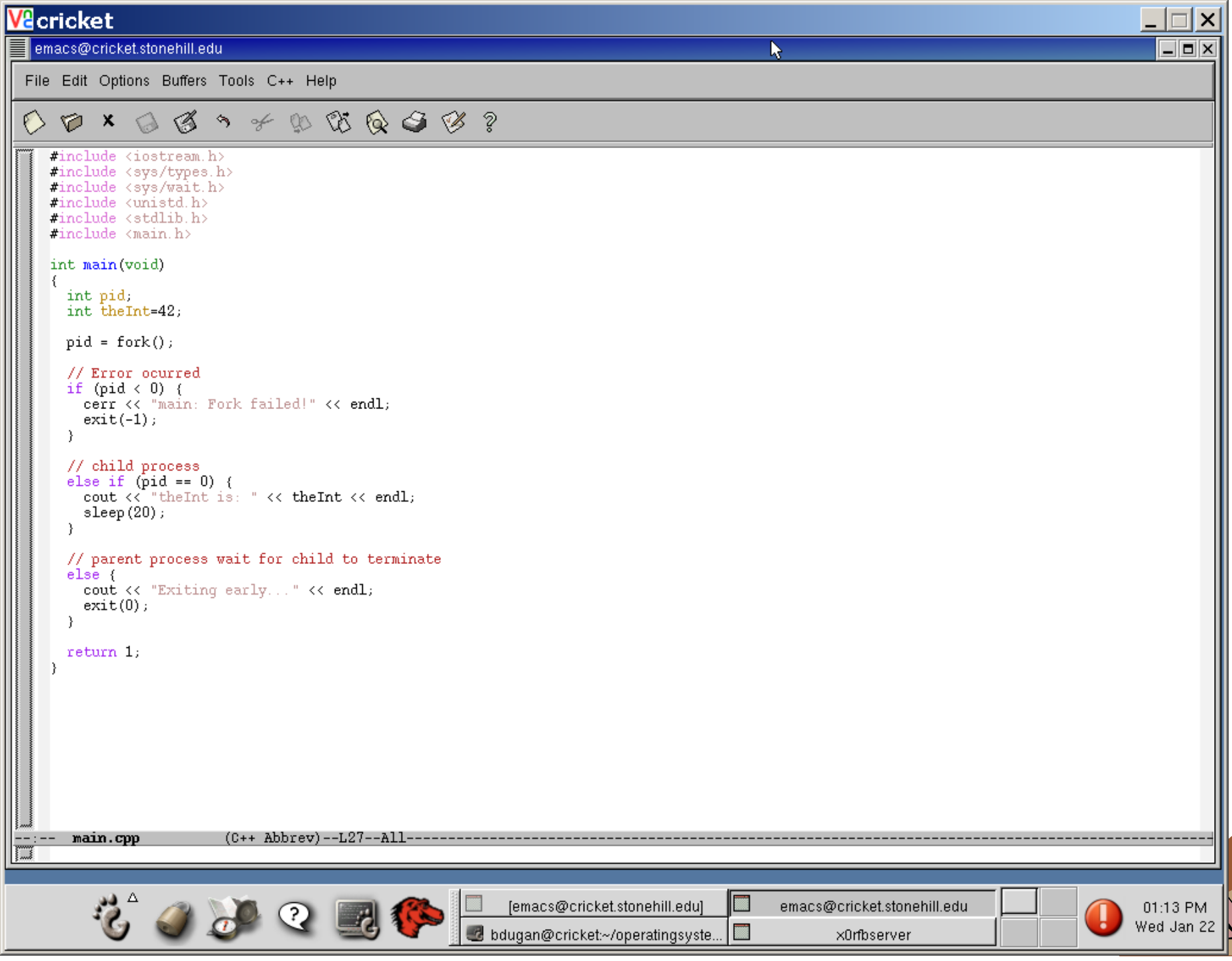
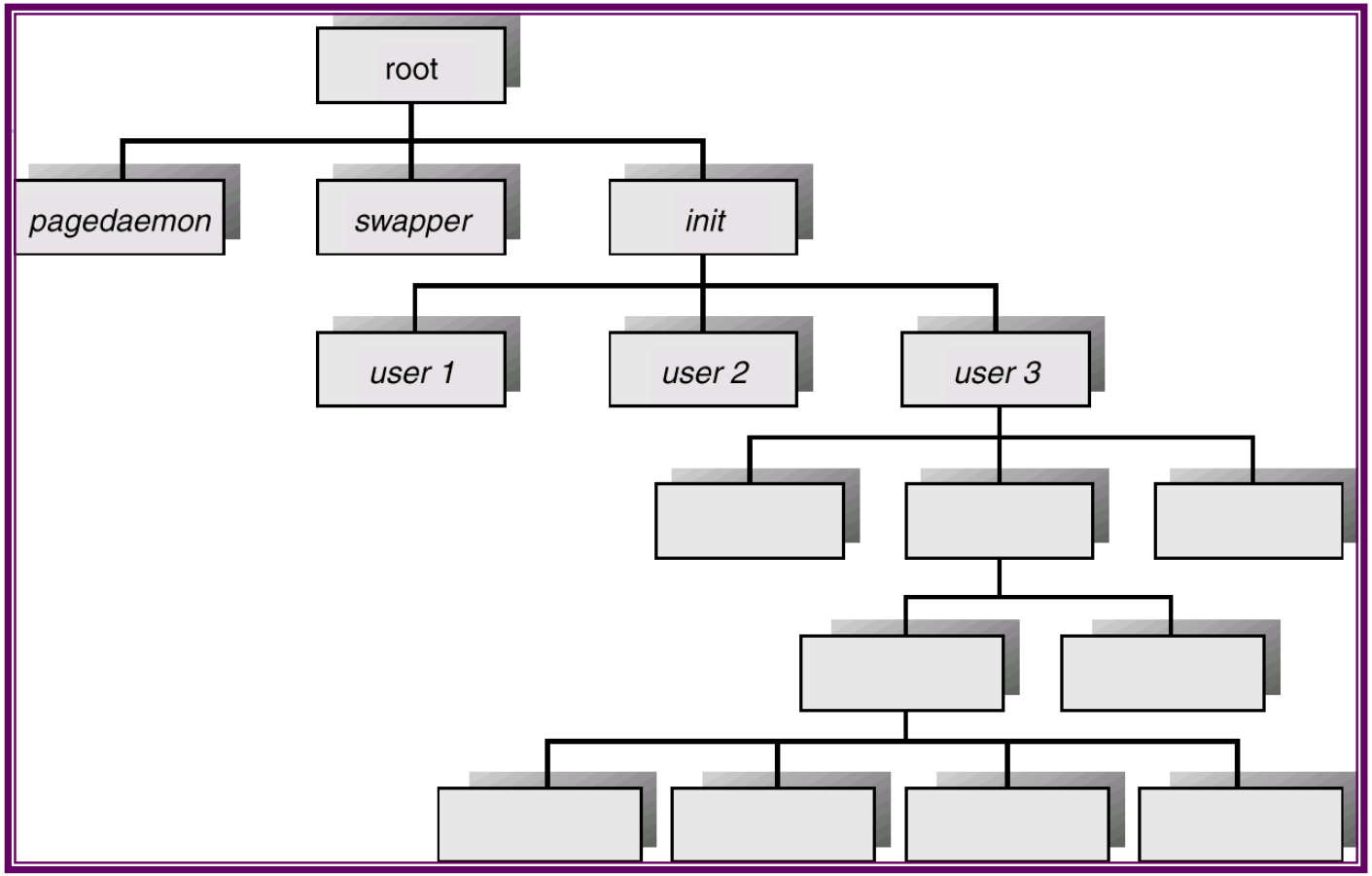

cricket
emacs@cricket.stonehill.edu
File Edit Options Buffers Tools C++ Help
#include <iostream.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <stdlib.h>
#include <main.h>
int main(void)
{
int pid;
int theInt=42;
pid = fork();
// Error ocurred
if (pid < 0) {
cerr << "main: Fork failed!" << endl;
exit(-1);
}
// child process
else if (pid == 0) {
cout << "theInt is: " << theInt << endl;
execlp("/bin/ls","ls",NULL);
}
// parent process wait for child to terminate
else {
wait(NULL);
cout << "Child complete" << endl;
exit(0);
}
return 1;
}
--:-- main.cpp (C++ Abbrev)--L17--All--
emacs@cricket.stonehill.edu
bdugan@cricket:~/rfb-0.1.2-i386-...
x0rfbserver
11:58 AM
Wed Jan 22


cricket
emacs@cricket.stonehill.edu
File Edit Options Buffers Tools C++ Help
#include <iostream.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <stdlib.h>
#include <main.h>
int main(void)
{
int pid;
int theInt=42;
pid = fork();
// Error ocurred
if (pid < 0) {
cerr << "main: Fork failed!" << endl;
exit(-1);
}
// child process
else if (pid == 0) {
cout << "theInt is: " << theInt << endl;
sleep(20);
}
// parent process wait for child to terminate
else {
cout << "Exiting early..." << endl;
exit(0);
}
return 1;
}
--:-- main.cpp (C++ Abbrev)--L27--All
[emacs@cricket.stonehill.edu]
emacs@cricket.stonehill.edu
bdugan@cricket:~/operatingsyste...
xOrfbserver
01:13 PM
Wed Jan 22

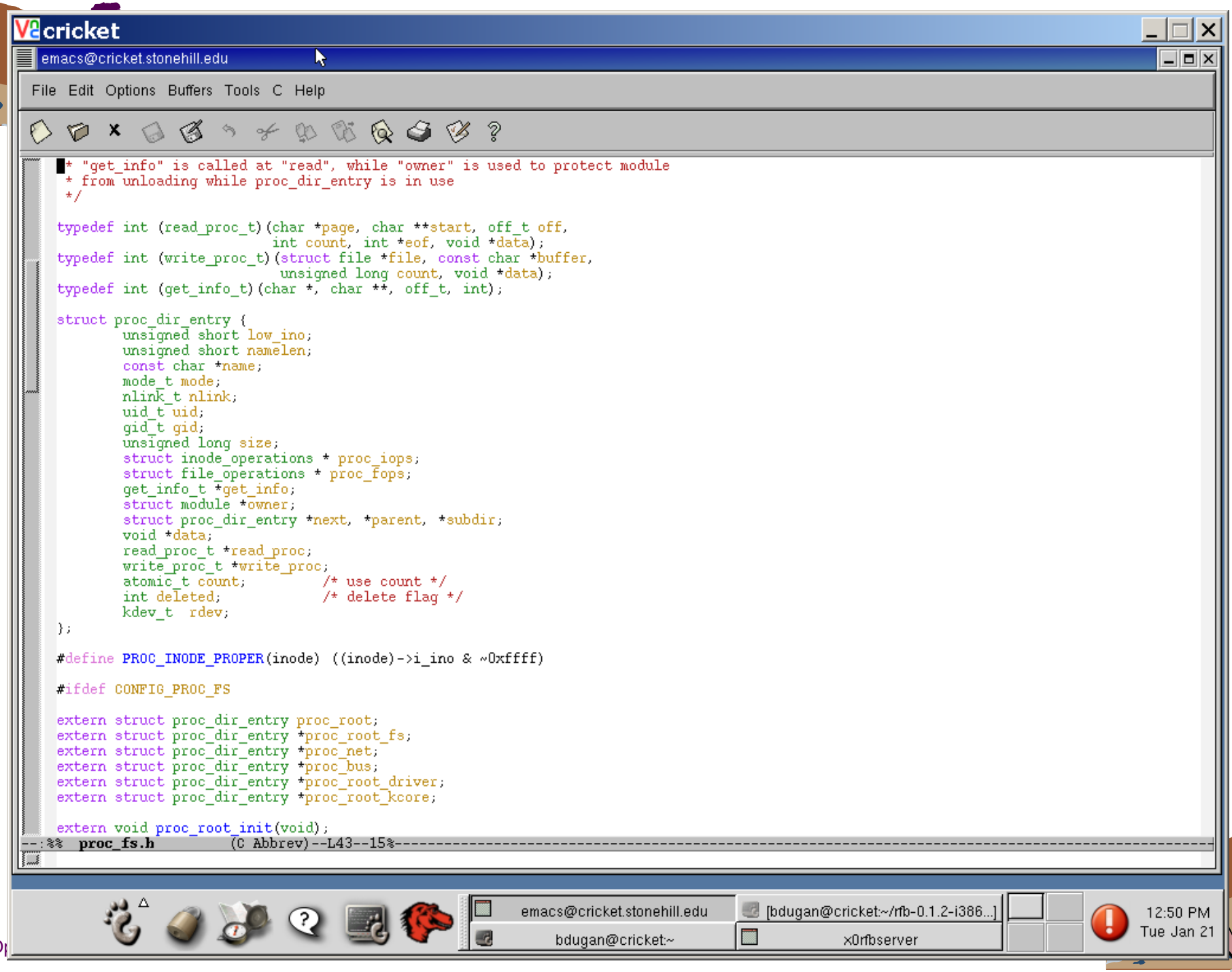
Processes Tree on a UNIX System



Sample Process Control Block

Process management	Memory management	File management
Registers	Pointer to text segment	Root directory
Program counter	Pointer to data segment	Working directory
Program status word	Pointer to stack segment	File descriptors
Stack pointer		User ID
Process state		Group ID
Priority		
Scheduling parameters		
Process ID		
Parent process		
Process group		
Signals		
Time when process started		
CPU time used		
Children's CPU time		
Time of next alarm		

Figure 2-4. Some of the fields of a typical process table entry.


cricket
emacs@cricket.stonehill.edu
File Edit Options Buffers Tools C Help
* "get_info" is called at "read", while "owner" is used to protect module
* from unloading while proc_dir_entry is in use
*/
typedef int (read_proc_t)(char *page, char **start, off_t off,
int count, int *eof, void *data);
typedef int (write_proc_t)(struct file *file, const char *buffer,
unsigned long count, void *data);
typedef int (get_info_t)(char *, char **, off_t, int);
struct proc_dir_entry {
unsigned short low_ino;
unsigned short namelen;
const char *name;
mode_t mode;
nlink_t nlink;
uid_t uid;
gid_t gid;
unsigned long size;
struct inode_operations * proc_iops;
struct file_operations * proc_fops;
get_info_t *get_info;
struct module *owner;
struct proc_dir_entry *next, *parent, *subdir;
void *data;
read_proc_t *read_proc;
write_proc_t *write_proc;
atomic_t count; /* use count */
int deleted; /* delete flag */
kdev_t rdev;
};
#define PROC_INODE_PROPER(inode) ((inode)->i_ino & ~0xffff)
#ifdef CONFIG_PROC_FS
extern struct proc_dir_entry proc_root;
extern struct proc_dir_entry *proc_root_fs;
extern struct proc_dir_entry *proc_net;
extern struct proc_dir_entry *proc_bus;
extern struct proc_dir_entry *proc_root_driver;
extern struct proc_dir_entry *proc_root_kcore;
extern void proc_root_init(void);
--:%% proc_fs.h (C Abbrev)--L43--15%
emacs@cricket.stonehill.edu
[bdugan@cricket:~/rfb-0.1.2-i386...]
bdugan@cricket:~
x0rfbserver
12:50 PM
Tue Jan 21