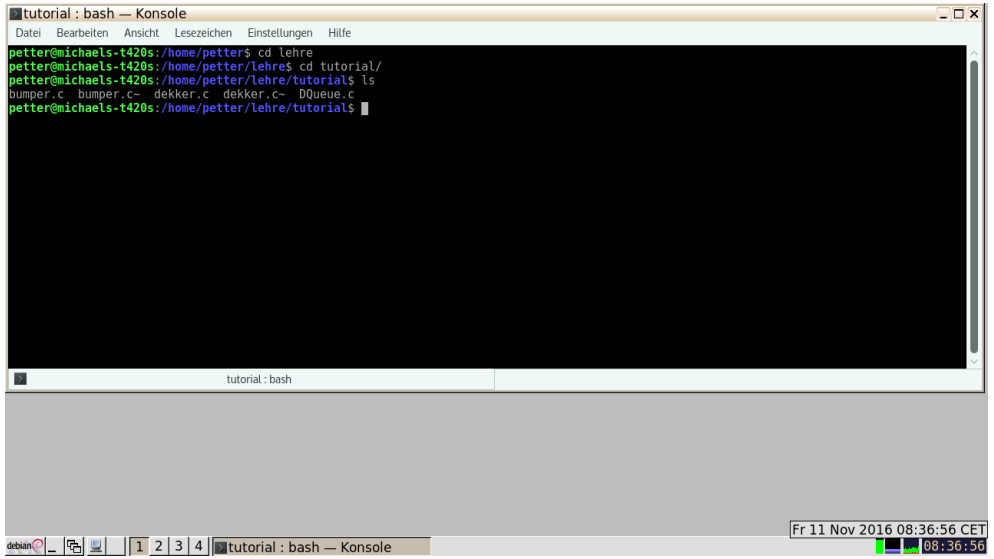
Script **generated by TTT**

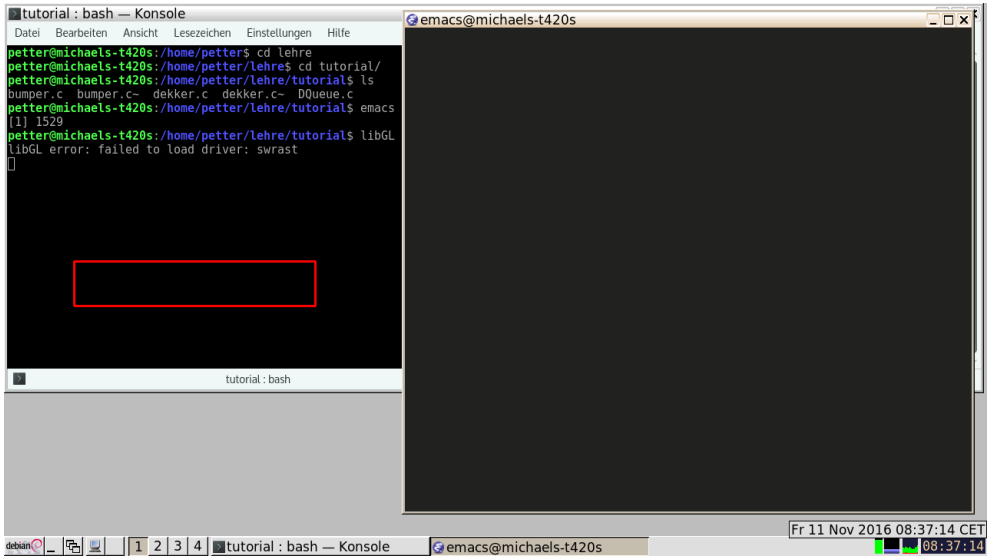
Title: Petter: Programmiersprachen_Uebung (11.11.2016)

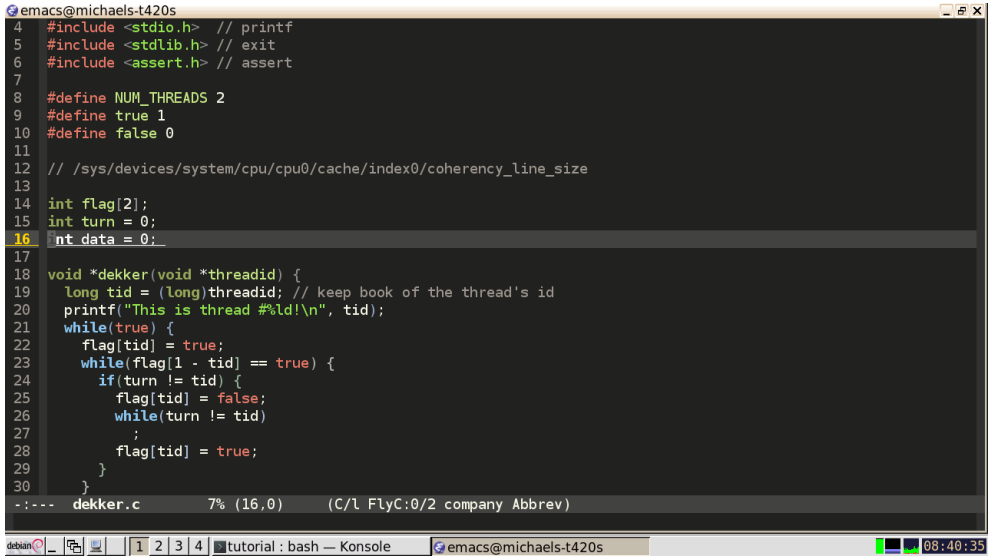
Date: Fri Nov 11 08:36:56 CET 2016

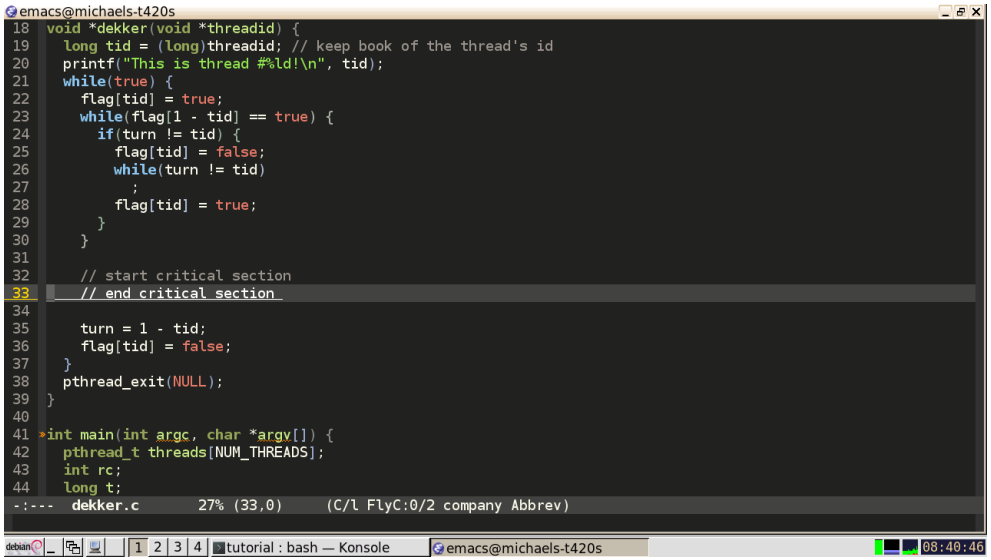
Duration: 79:53 min

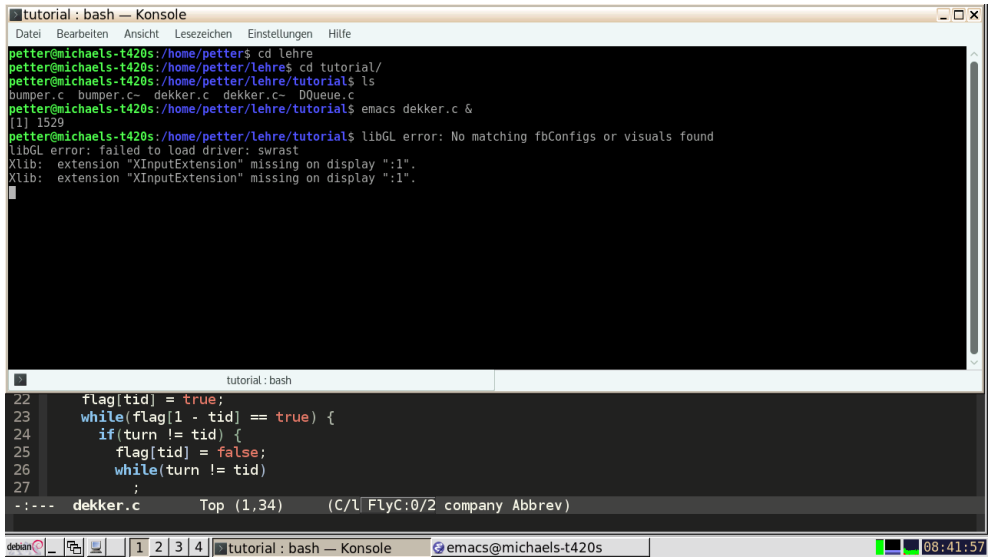
Pages: 81

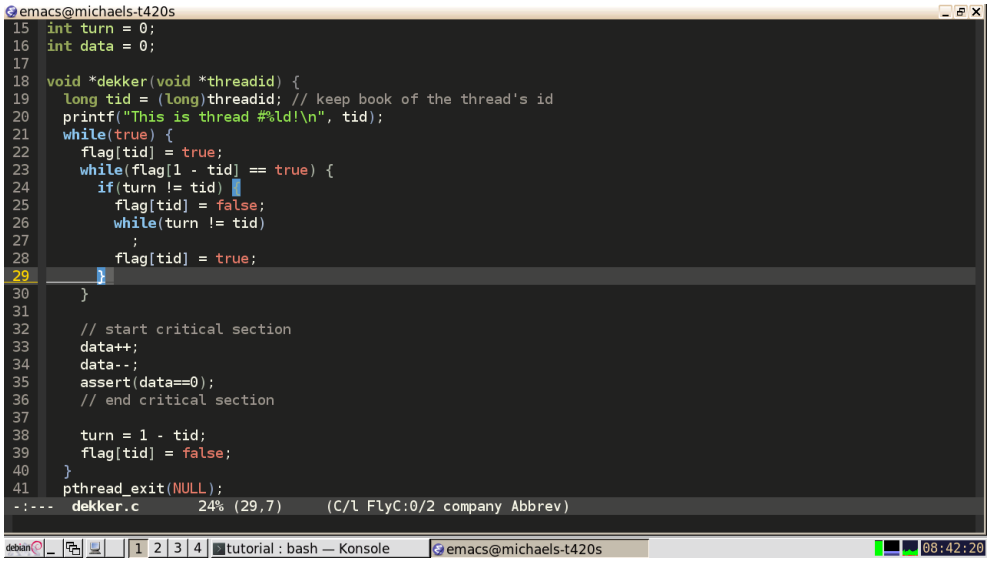


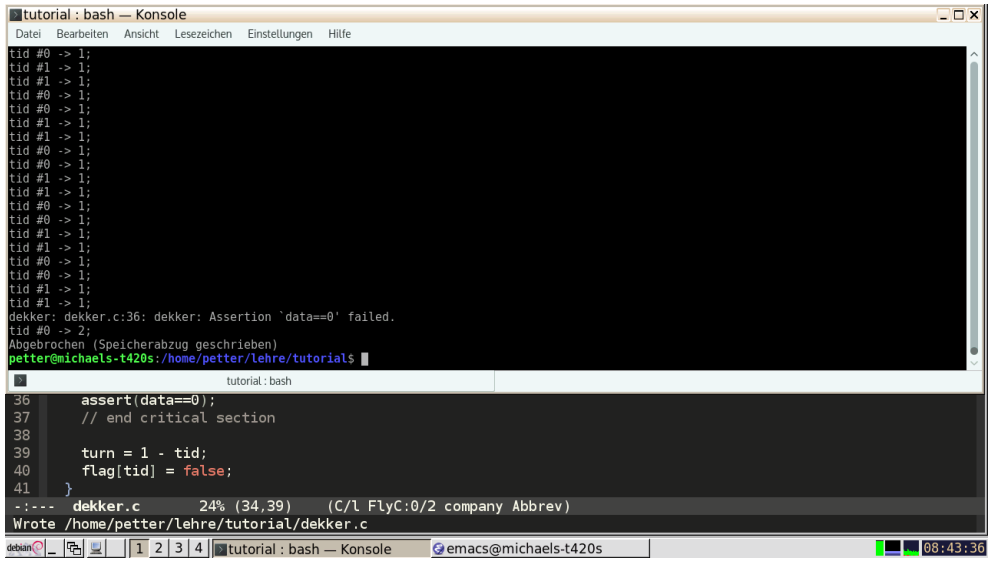


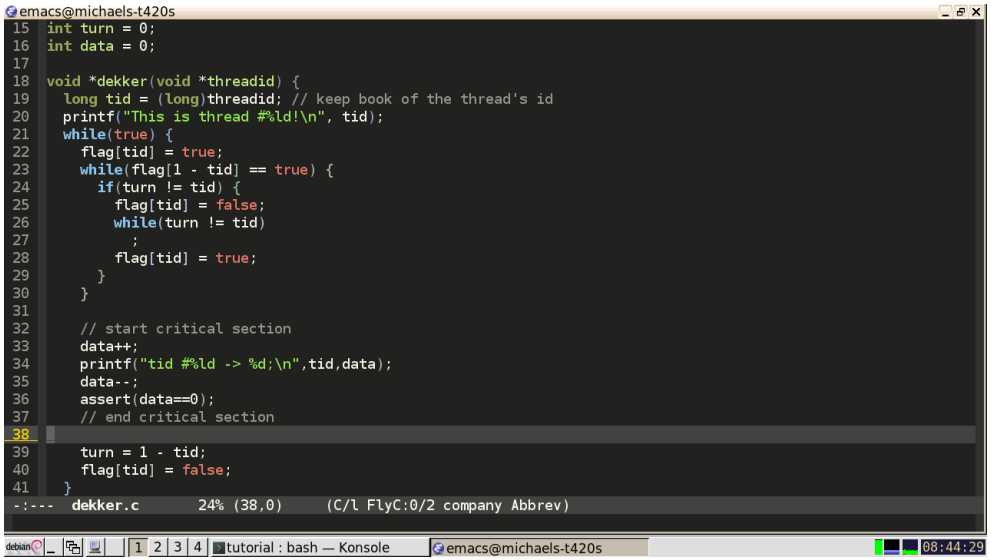


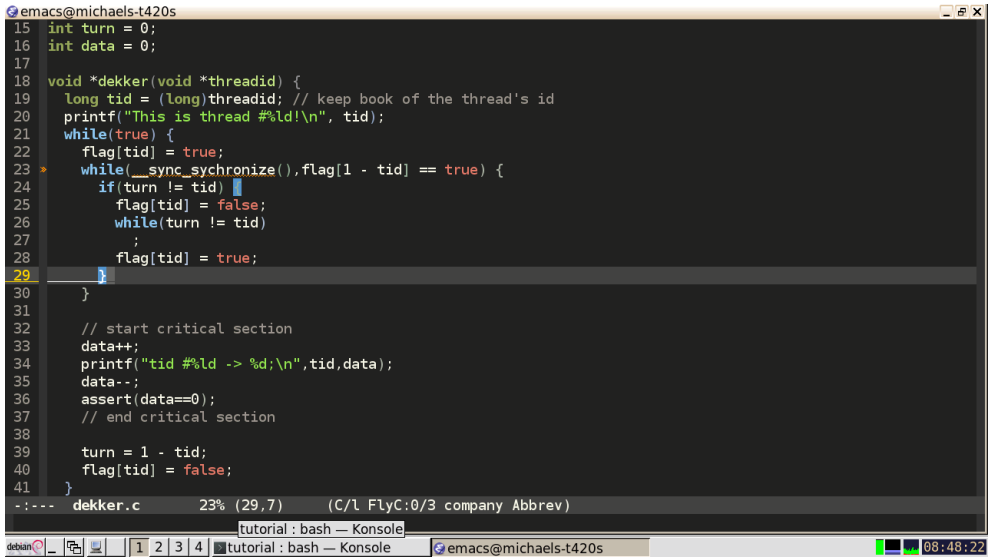

emacs@michaels-t420s
18 void *dekker(void *threadid) {
19 long tid = (long)threadid; // keep book of the thread's id
20 printf("This is thread #%ld!\n", tid);
21 while(true) {
22 flag[tid] = true;
23 while(flag[1 - tid] == true) {
24 if(turn != tid) {
25 flag[tid] = false;
26 while(turn != tid)
27 ;
28 flag[tid] = true;
29 }
30 }
31
32 // start critical section
33 // end critical section
34
35 turn = 1 - tid;
36 flag[tid] = false;
37 }
38 pthread_exit(NULL);
39 }
40
41 int main(int argc, char *argv[]) {
42 pthread_t threads[NUM_THREADS];
43 int rc;
44 long t;
-:--- dekker.c 27% (33,0) (C/l FlyC:0/2 company Abbrev)
1 2 3 4 tutorial : bash — Konsole emacs@michaels-t420s 08:40:46

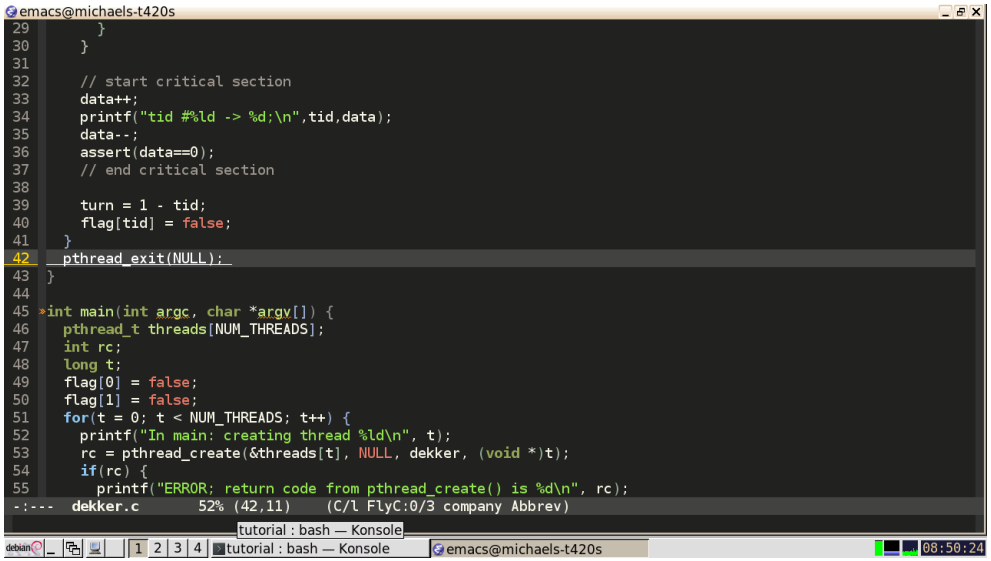

tutorial : bash — Konsole
Datei Bearbeiten Ansicht Lesezeichen Einstellungen Hilfe
petter@michaels-t420s:/home/petter$ cd lehre
petter@michaels-t420s:/home/petter/lehre$ cd tutorial/
petter@michaels-t420s:/home/petter/lehre/tutorial$ ls
bumper.c bumper.c~ dekker.c dekker.c~ DQueue.c
petter@michaels-t420s:/home/petter/lehre/tutorial$ emacs dekker.c &
[1] 1529
petter@michaels-t420s:/home/petter/lehre/tutorial$ libGL error: No matching fbConfigs or visuals found
libGL error: failed to load driver: swrast
Xlib: extension "XInputExtension" missing on display ":1".
Xlib: extension "XInputExtension" missing on display ":1".
tutorial : bash
22 flag[tid] = true;
23 while(flag[1 - tid] == true) {
24 if(turn != tid) {
25 flag[tid] = false;
26 while(turn != tid)
27 ;
-:--- dekker.c Top (1,34) (C/l FlyC:0/2 company Abbrev)
1 2 3 4 tutorial : bash — Konsole emacs@michaels-t420s 08:41:57

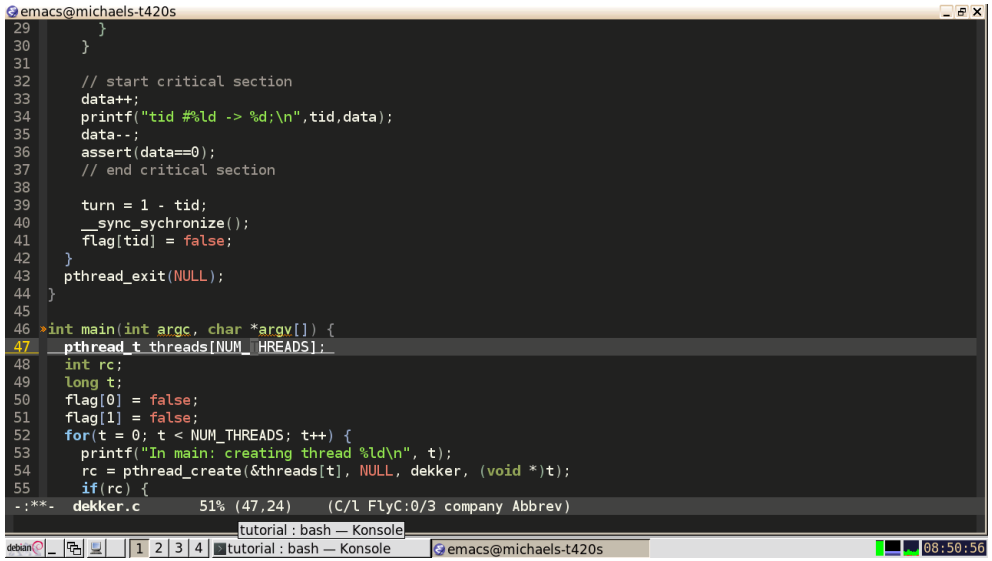

emacs@michaels-t420s
15 int turn = 0;
16 int data = 0;
17
18 void *dekker(void *threadid) {
19 long tid = (long)threadid; // keep book of the thread's id
20 printf("This is thread #%ld!\n", tid);
21 while(true) {
22 flag[tid] = true;
23 while(flag[1 - tid] == true) {
24 if(turn != tid) {
25 flag[tid] = false;
26 while(turn != tid)
27 ;
28 flag[tid] = true;
29 }
30 }
31
32 // start critical section
33 data++;
34 data--;
35 assert(data==0);
36 // end critical section
37
38 turn = 1 - tid;
39 flag[tid] = false;
40 }
41 pthread_exit(NULL);
-:--- dekker.c 24% (29,7) (C/l FlyC:0/2 company Abbrev)
1 2 3 4 tutorial : bash — Konsole emacs@michaels-t420s 08:42:20

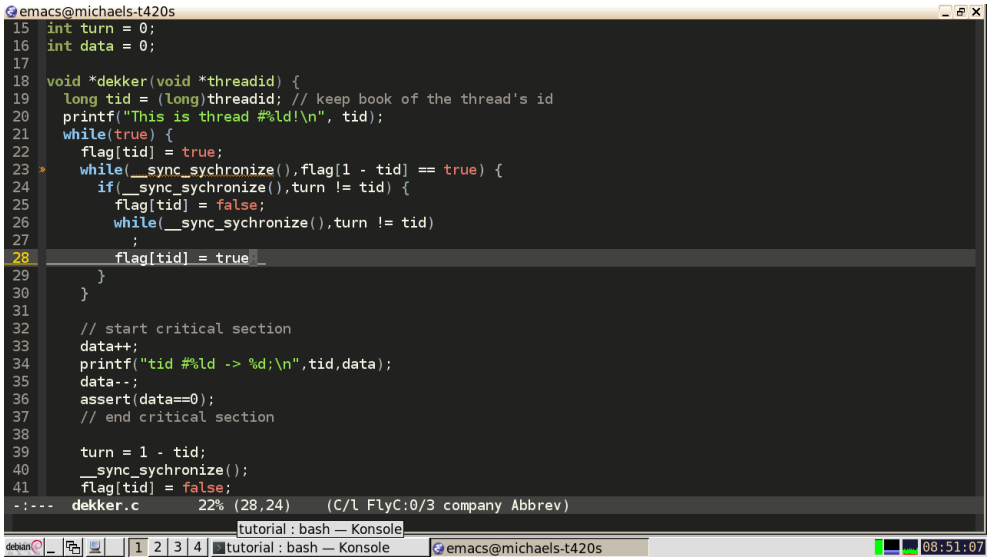

tutorial : bash — Konsole
Datei Bearbeiten Ansicht Lesezeichen Einstellungen Hilfe
tid #0 -> 1;
tid #1 -> 1;
tid #1 -> 1;
tid #0 -> 1;
tid #0 -> 1;
tid #1 -> 1;
tid #1 -> 1;
tid #0 -> 1;
tid #0 -> 1;
tid #1 -> 1;
tid #1 -> 1;
tid #0 -> 1;
tid #0 -> 1;
tid #1 -> 1;
tid #1 -> 1;
tid #0 -> 1;
tid #0 -> 1;
tid #1 -> 1;
tid #1 -> 1;
dekker: dekker.c:36: dekker: Assertion `data==0' failed.
tid #0 -> 2;
Abgebrochen (Speicherabzug geschrieben)
petter@michaels-t420s:/home/petter/lehre/tutorial$
tutorial : bash
36 assert(data==0);
37 // end critical section
38
39 turn = 1 - tid;
40 flag[tid] = false;
41 }
-:--- dekker.c 24% (34,39) (C/l FlyC:0/2 company Abbrev)
Wrote /home/petter/lehre/tutorial/dekker.c
1 2 3 4 tutorial : bash — Konsole emacs@michaels-t420s 08:43:36

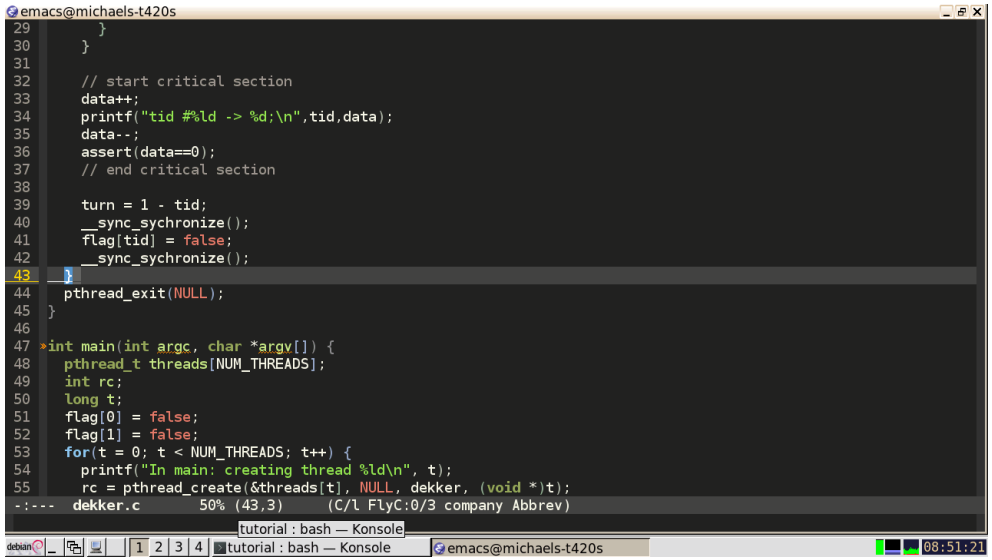

emacs@michaels-t420s
int turn = 0;
int data = 0;
void *dekker(void *threadid) {
long tid = (long)threadid; // keep book of the thread's id
printf("This is thread #%ld!\n", tid);
while(true) {
flag[tid] = true;
while(flag[1 - tid] == true) {
if(turn != tid) {
flag[tid] = false;
while(turn != tid)
;
flag[tid] = true;
}
}
// start critical section
data++;
printf("tid #%ld -> %d;\n",tid,data);
data--;
assert(data==0);
// end critical section
turn = 1 - tid;
flag[tid] = false;
}
-:--- dekker.c 24% (38,0) (C/l FlyC:0/2 company Abbrev)
tutorial : bash — Konsole
emacs@michaels-t420s
08:44:29

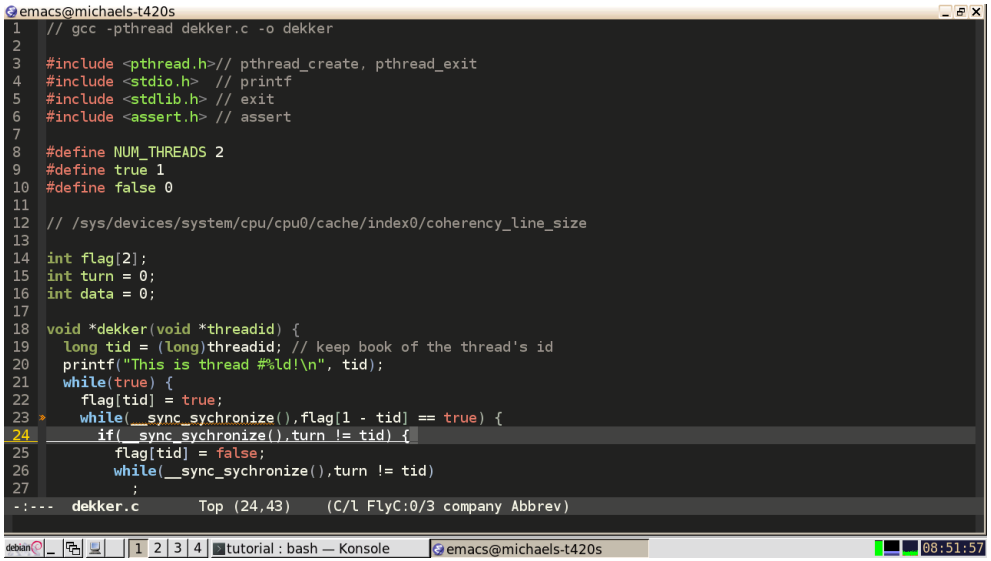

emacs@michaels-t420s
int turn = 0;
int data = 0;
void *dekker(void *threadid) {
long tid = (long)threadid; // keep book of the thread's id
printf("This is thread #%ld!\n", tid);
while(true) {
flag[tid] = true;
while(__sync_sychronize(),flag[1 - tid] == true) {
if(turn != tid) {
flag[tid] = false;
while(turn != tid)
;
flag[tid] = true;
}
}
// start critical section
data++;
printf("tid #%ld -> %d;\n",tid,data);
data--;
assert(data==0);
// end critical section
turn = 1 - tid;
flag[tid] = false;
}
-:--- dekker.c 23% (29,7) (C/l FlyC:0/3 company Abbrev)
tutorial : bash — Konsole
tutorial : bash — Konsole
emacs@michaels-t420s
08:48:22

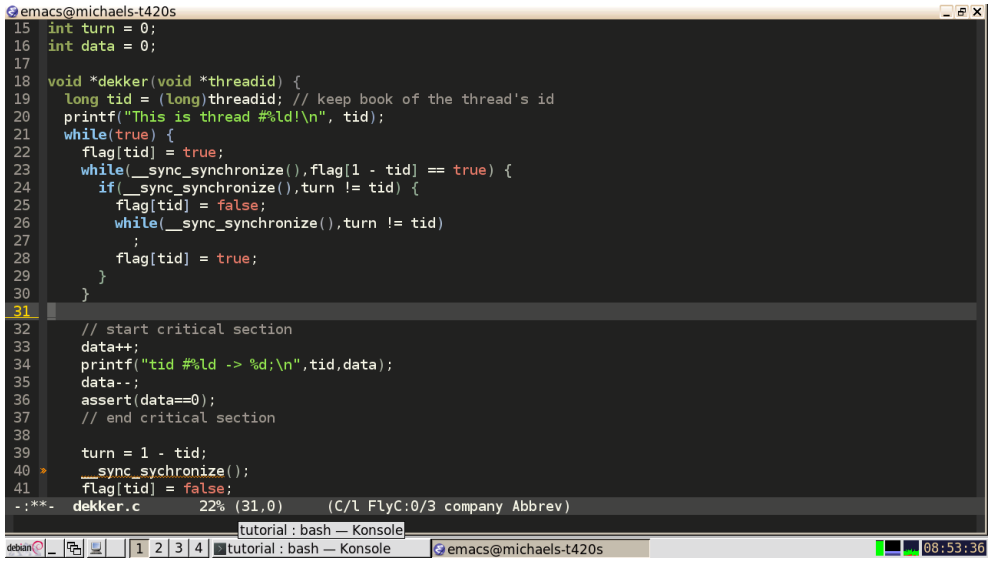

emacs@michaels-t420s
}
}
// start critical section
data++;
printf("tid #%ld -> %d;\n",tid,data);
data--;
assert(data==0);
// end critical section
turn = 1 - tid;
flag[tid] = false;
}
pthread_exit(NULL);
}
int main(int argc, char *argv[]) {
pthread_t threads[NUM_THREADS];
int rc;
long t;
flag[0] = false;
flag[1] = false;
for(t = 0; t < NUM_THREADS; t++) {
printf("In main: creating thread %ld\n", t);
rc = pthread_create(&threads[t], NULL, dekker, (void *)t);
if(rc) {
printf("ERROR; return code from pthread_create() is %d\n", rc);
-:--- dekker.c 52% (42,11) (C/l FlyC:0/3 company Abbrev)
tutorial : bash — Konsole
tutorial : bash — Konsole
emacs@michaels-t420s
08:50:24

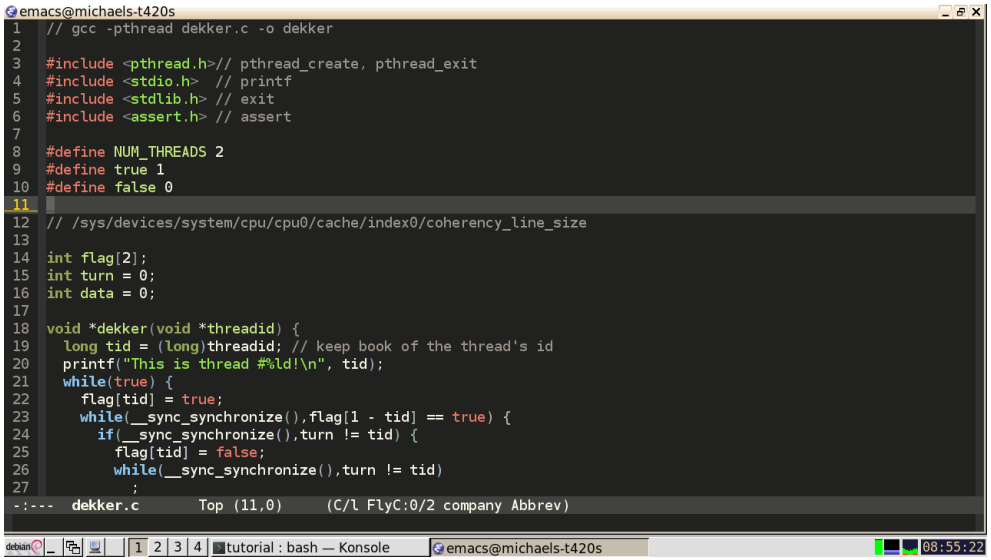

emacs@michaels-t420s
}
}
// start critical section
data++;
printf("tid #%ld -> %d;\n",tid,data);
data--;
assert(data==0);
// end critical section
turn = 1 - tid;
__sync_sychronize();
flag[tid] = false;
}
pthread_exit(NULL);
}
int main(int argc, char *argv[]) {
pthread_t threads[NUM_THREADS];
int rc;
long t;
flag[0] = false;
flag[1] = false;
for(t = 0; t < NUM_THREADS; t++) {
printf("In main: creating thread %ld\n", t);
rc = pthread_create(&threads[t], NULL, dekker, (void *)t);
if(rc) {
-:**- dekker.c 51% (47,24) (C/l FlyC:0/3 company Abbrev)
tutorial : bash — Konsole
tutorial : bash — Konsole
emacs@michaels-t420s
08:50:56

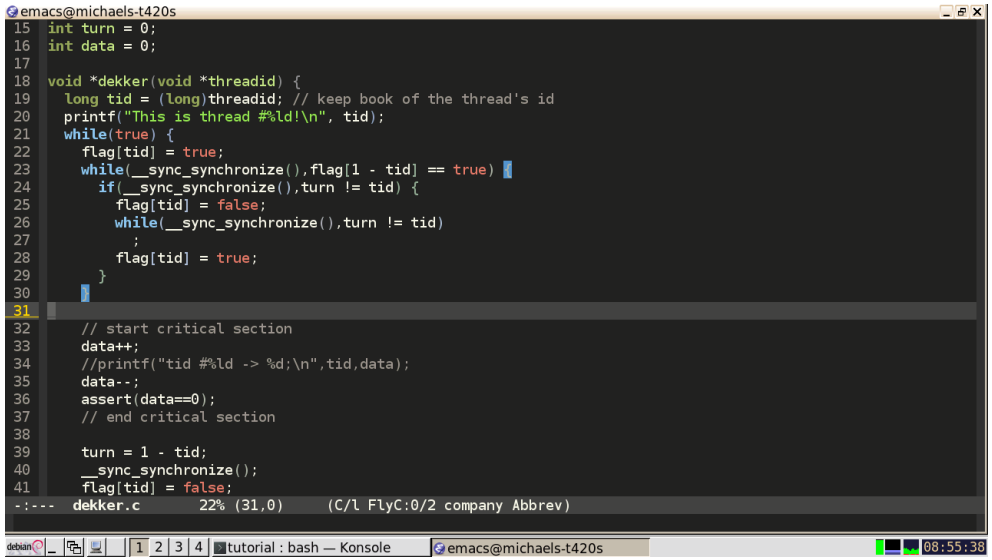

emacs@michaels-t420s
int turn = 0;
int data = 0;
void *dekker(void *threadid) {
long tid = (long)threadid; // keep book of the thread's id
printf("This is thread #%ld!\n", tid);
while(true) {
flag[tid] = true;
while(__sync_sychronize(),flag[1 - tid] == true) {
if(__sync_sychronize(),turn != tid) {
flag[tid] = false;
while(__sync_sychronize(),turn != tid)
;
flag[tid] = true
}
}
// start critical section
data++;
printf("tid #%ld -> %d;\n",tid,data);
data--;
assert(data==0);
// end critical section
turn = 1 - tid;
__sync_sychronize();
flag[tid] = false;
-:--- dekker.c 22% (28,24) (C/l FlyC:0/3 company Abbrev)
tutorial : bash — Konsole
1 2 3 4
tutorial : bash — Konsole
emacs@michaels-t420s
08:51:07

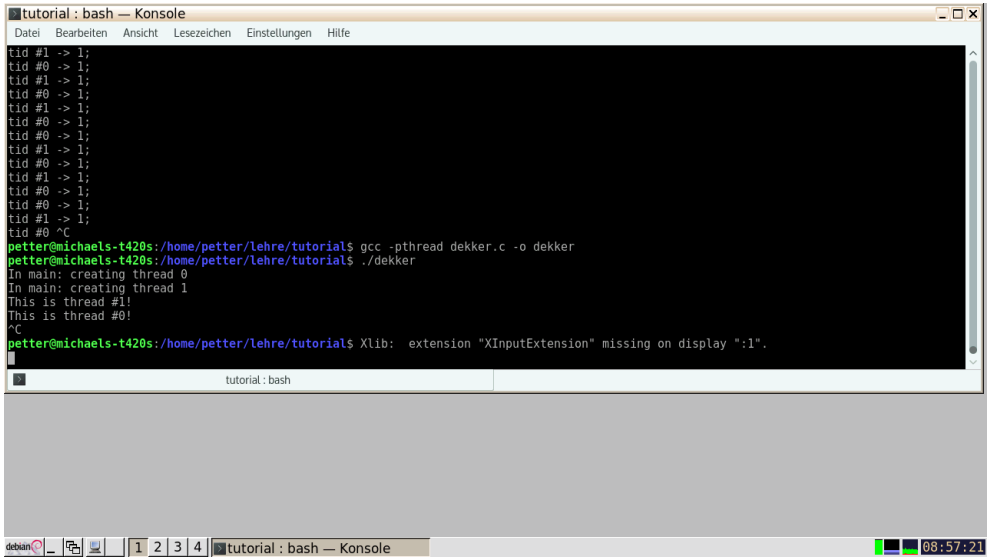

emacs@michaels-t420s
}
}
// start critical section
data++;
printf("tid #%ld -> %d;\n",tid,data);
data--;
assert(data==0);
// end critical section
turn = 1 - tid;
__sync_sychronize();
flag[tid] = false;
__sync_sychronize();
}
pthread_exit(NULL);
}
int main(int argc, char *argv[]) {
pthread_t threads[NUM_THREADS];
int rc;
long t;
flag[0] = false;
flag[1] = false;
for(t = 0; t < NUM_THREADS; t++) {
printf("In main: creating thread %ld\n", t);
rc = pthread_create(&threads[t], NULL, dekker, (void *)t);
-:--- dekker.c 50% (43,3) (C/l FlyC:0/3 company Abbrev)
tutorial : bash — Konsole
1 2 3 4
tutorial : bash — Konsole
emacs@michaels-t420s
08:51:21

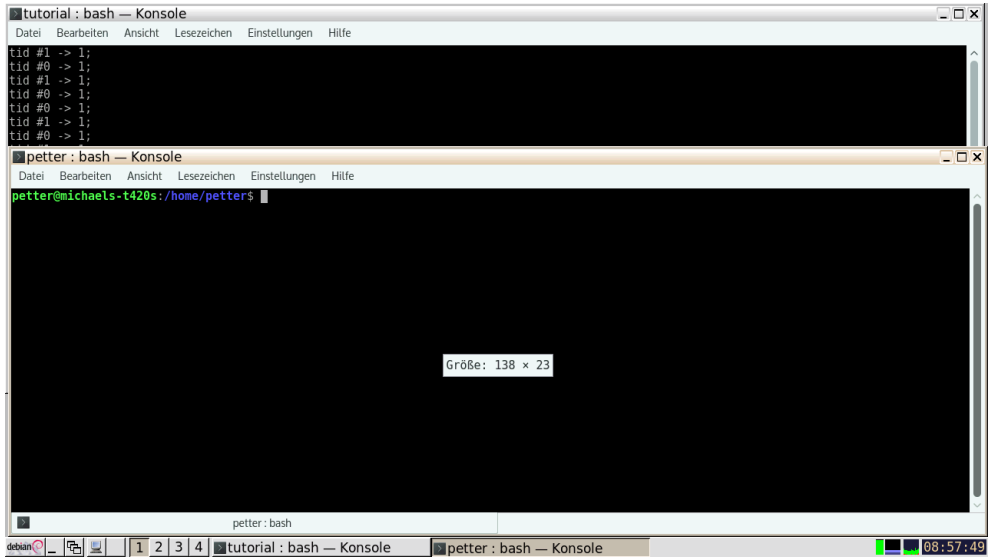

emacs@michaels-t420s
// gcc -pthread dekker.c -o dekker
#include <pthread.h>// pthread_create, pthread_exit
#include <stdio.h> // printf
#include <stdlib.h> // exit
#include <assert.h> // assert
#define NUM_THREADS 2
#define true 1
#define false 0
// /sys/devices/system/cpu/cpu0/cache/index0/coherency_line_size
int flag[2];
int turn = 0;
int data = 0;
void *dekker(void *threadid) {
long tid = (long)threadid; // keep book of the thread's id
printf("This is thread #%ld!\n", tid);
while(true) {
flag[tid] = true;
while(__sync_sychronize(),flag[1 - tid] == true) {
if(__sync_sychronize(),turn != tid) {
flag[tid] = false;
while(__sync_sychronize(),turn != tid)
;
-:--- dekker.c Top (24,43) (C/l FlyC:0/3 company Abbrev)
1 2 3 4
tutorial : bash — Konsole
emacs@michaels-t420s
08:51:57

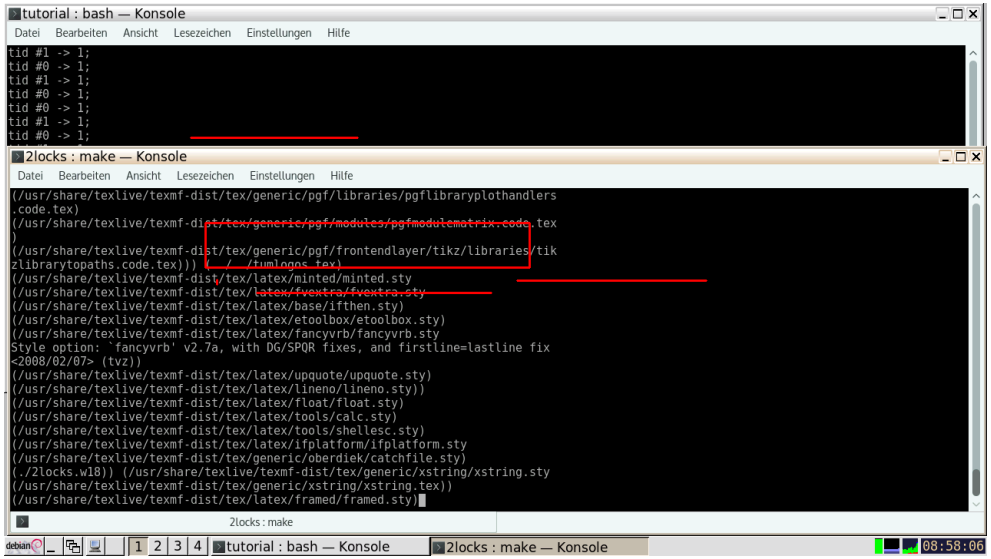

emacs@michaels-t420s
int turn = 0;
int data = 0;
void *dekker(void *threadid) {
long tid = (long)threadid; // keep book of the thread's id
printf("This is thread #%ld!\n", tid);
while(true) {
flag[tid] = true;
while(__sync_synchronize(),flag[1 - tid] == true) {
if(__sync_synchronize(),turn != tid) {
flag[tid] = false;
while(__sync_synchronize(),turn != tid)
;
flag[tid] = true;
}
}
// start critical section
data++;
printf("tid #%ld -> %d;\n",tid,data);
data--;
assert(data==0);
// end critical section
turn = 1 - tid;
__sync_sychronize();
flag[tid] = false;
-:**- dekker.c 22% (31,0) (C/l FlyC:0/3 company Abbrev)
tutorial : bash — Konsole
1 2 3 4
tutorial : bash — Konsole
emacs@michaels-t420s
08:53:36

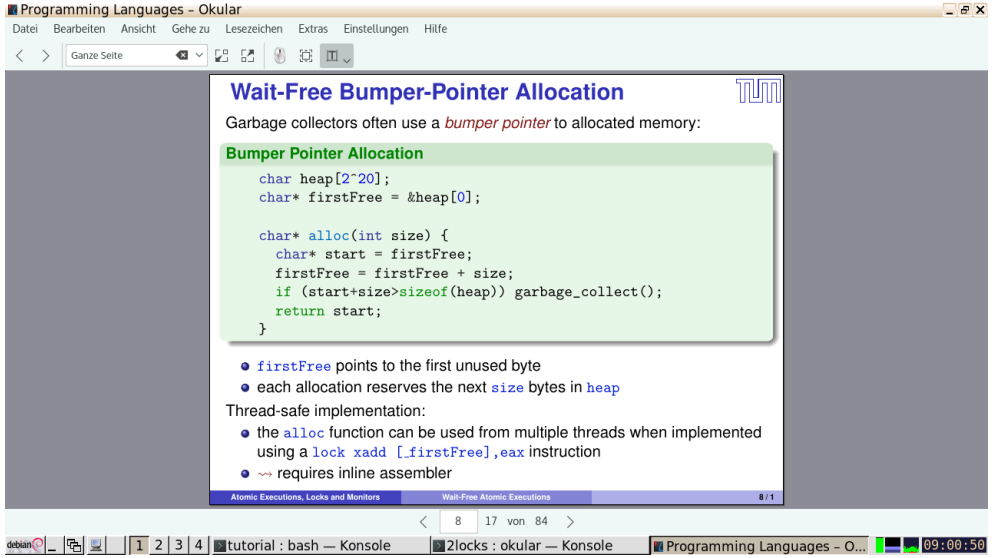

emacs@michaels-t420s
// gcc -pthread dekker.c -o dekker
#include <pthread.h>// pthread_create, pthread_exit
#include <stdio.h> // printf
#include <stdlib.h> // exit
#include <assert.h> // assert
#define NUM_THREADS 2
#define true 1
#define false 0
// /sys/devices/system/cpu/cpu0/cache/index0/coherency_line_size
int flag[2];
int turn = 0;
int data = 0;
void *dekker(void *threadid) {
long tid = (long)threadid; // keep book of the thread's id
printf("This is thread #%ld!\n", tid);
while(true) {
flag[tid] = true;
while(__sync_synchronize(),flag[1 - tid] == true) {
if(__sync_synchronize(),turn != tid) {
flag[tid] = false;
while(__sync_synchronize(),turn != tid)
;
-:--- dekker.c Top (11,0) (C/l FlyC:0/2 company Abbrev)
tutorial : bash — Konsole
emacs@michaels-t420s
08:55:22

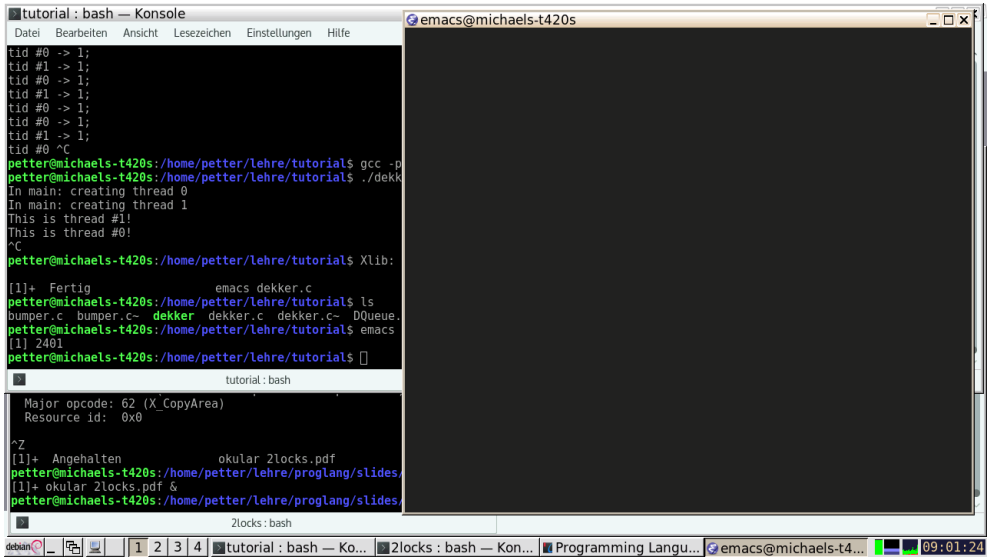

emacs@michaels-t420s
int turn = 0;
int data = 0;
void *dekker(void *threadid) {
long tid = (long)threadid; // keep book of the thread's id
printf("This is thread #%ld!\n", tid);
while(true) {
flag[tid] = true;
while(__sync_synchronize(),flag[1 - tid] == true) {
if(__sync_synchronize(),turn != tid) {
flag[tid] = false;
while(__sync_synchronize(),turn != tid)
;
flag[tid] = true;
}
}
// start critical section
data++;
//printf("tid #%ld -> %d;\n",tid,data);
data--;
assert(data==0);
// end critical section
turn = 1 - tid;
__sync_synchronize();
flag[tid] = false;
-:--- dekker.c 22% (31,0) (C/l FlyC:0/2 company Abbrev)
tutorial : bash — Konsole
emacs@michaels-t420s
08:55:38

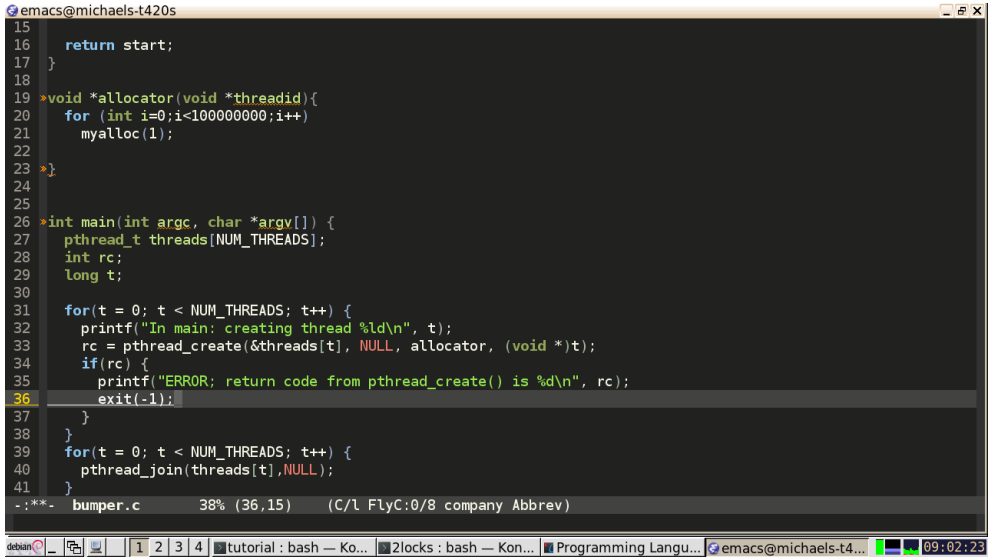

tutorial : bash — Konsole
Datei Bearbeiten Ansicht Lesezeichen Einstellungen Hilfe
tid #1 -> 1;
tid #0 -> 1;
tid #1 -> 1;
tid #0 -> 1;
tid #1 -> 1;
tid #0 -> 1;
tid #0 -> 1;
tid #1 -> 1;
tid #0 -> 1;
tid #1 -> 1;
tid #0 -> 1;
tid #0 -> 1;
tid #1 -> 1;
tid #0 ^C
petter@michaels-t420s:/home/petter/lehre/tutorial$ gcc -pthread dekker.c -o dekker
petter@michaels-t420s:/home/petter/lehre/tutorial$ ./dekker
In main: creating thread 0
In main: creating thread 1
This is thread #1!
This is thread #0!
^C
petter@michaels-t420s:/home/petter/lehre/tutorial$ Xlib: extension "XInputExtension" missing on display ":1".
tutorial : bash
tutorial : bash — Konsole
08:57:21

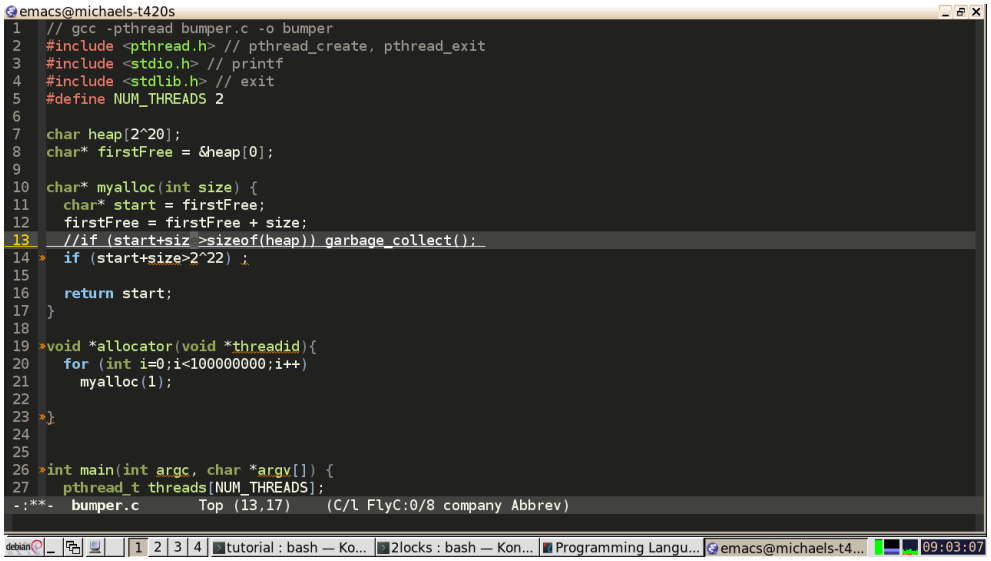

tutorial : bash — Konsole
Datei Bearbeiten Ansicht Lesezeichen Einstellungen Hilfe
tid #1 -> 1;
tid #0 -> 1;
tid #1 -> 1;
tid #0 -> 1;
tid #0 -> 1;
tid #1 -> 1;
tid #0 -> 1;
petter : bash — Konsole
Datei Bearbeiten Ansicht Lesezeichen Einstellungen Hilfe
petter@michaels-t420s:/home/petter$
Größe: 138 × 23
petter : bash
tutorial : bash — Konsole
petter : bash — Konsole
08:57:49

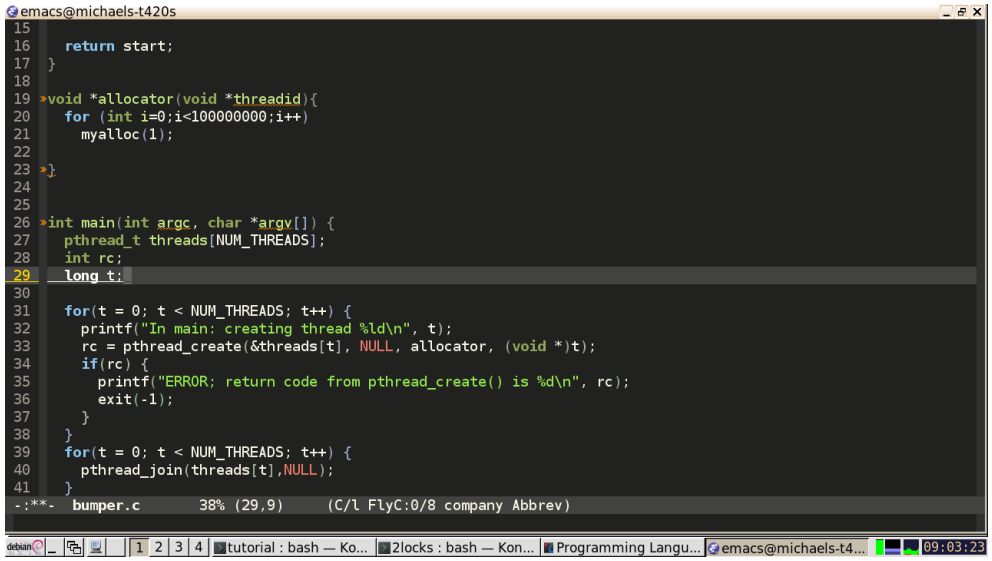

tutorial : bash — Konsole
Datei Bearbeiten Ansicht Lesezeichen Einstellungen Hilfe
tid #1 -> 1;
tid #0 -> 1;
tid #1 -> 1;
tid #0 -> 1;
tid #0 -> 1;
tid #1 -> 1;
tid #0 -> 1;
2locks : make — Konsole
Datei Bearbeiten Ansicht Lesezeichen Einstellungen Hilfe
(/usr/share/texlive/texmf-dist/tex/generic/pgf/libraries/pgflibraryplothandlers
.code.tex)
(/usr/share/texlive/texmf-dist/tex/generic/pgf/modules/pgfmodulematrix.code.tex
)
(/usr/share/texlive/texmf-dist/tex/generic/pgf/frontendlayer/tikz/libraries/tik
zlibrarytopaths.code.tex))) (./_tumlogos.tex)
(/usr/share/texlive/texmf-dist/tex/latex/minted/minted.sty
(/usr/share/texlive/texmf-dist/tex/latex/fvextra/fvextra.sty
(/usr/share/texlive/texmf-dist/tex/latex/base/ifthen.sty)
(/usr/share/texlive/texmf-dist/tex/latex/etoolbox/etoolbox.sty)
(/usr/share/texlive/texmf-dist/tex/latex/fancyvrb/fancyvrb.sty
Style option: `fancyvrb' v2.7a, with DG/SPQR fixes, and firstline=lastline fix
<2008/02/07> (tvz))
(/usr/share/texlive/texmf-dist/tex/latex/upquote/upquote.sty)
(/usr/share/texlive/texmf-dist/tex/latex/lineno/lineno.sty))
(/usr/share/texlive/texmf-dist/tex/latex/float/float.sty)
(/usr/share/texlive/texmf-dist/tex/latex/tools/calc.sty)
(/usr/share/texlive/texmf-dist/tex/latex/tools/shellesc.sty)
(/usr/share/texlive/texmf-dist/tex/latex/ifplatform/ifplatform.sty
(/usr/share/texlive/texmf-dist/tex/generic/oberdiek/catchfile.sty)
(./2locks.w18)) (/usr/share/texlive/texmf-dist/tex/generic/xstring/xstring.sty
(/usr/share/texlive/texmf-dist/tex/generic/xstring/xstring.tex))
(/usr/share/texlive/texmf-dist/tex/latex/framed/framed.sty)
2locks : make
1 2 3 4
tutorial : bash — Konsole
2locks : make — Konsole
08:58:06

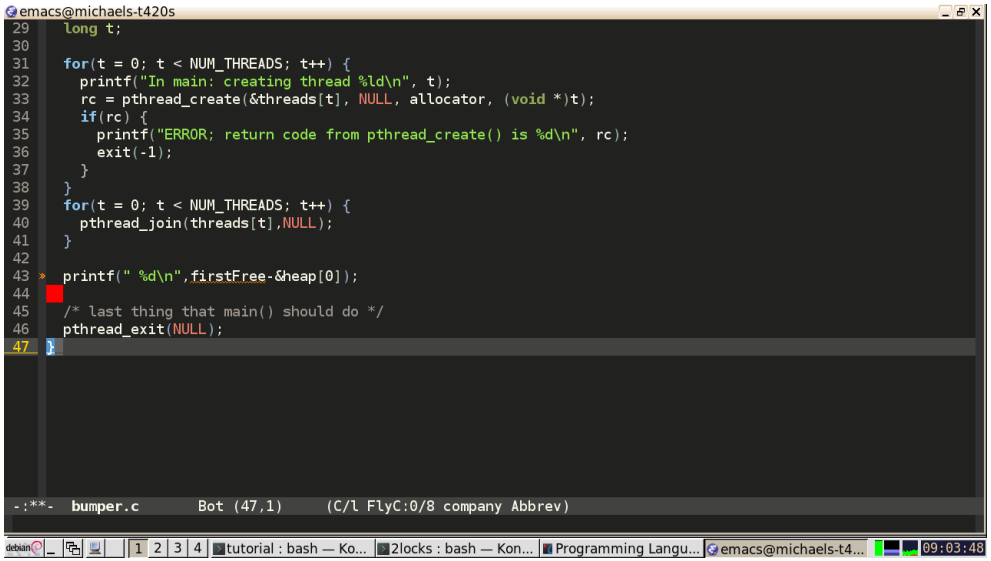

Programming Languages – Okular
Datei Bearbeiten Ansicht Gehe zu Lesezeichen Extras Einstellungen Hilfe
Ganze Seite
Wait-Free Bumper-Pointer Allocation
Garbage collectors often use a bumper pointer to allocated memory:
Bumper Pointer Allocation
char heap[2^20];
char* firstFree = &heap[0];
char* alloc(int size) {
char* start = firstFree;
firstFree = firstFree + size;
if (start+size>sizeof(heap)) garbage_collect();
return start;
}
firstFree points to the first unused byte
each allocation reserves the next size bytes in heap
Thread-safe implementation:
the alloc function can be used from multiple threads when implemented using a lock xadd [_firstFree],eax instruction
requires inline assembler
Atomic Executions, Locks and Monitors
Wait-Free Atomic Executions
8 / 1
8 17 von 84
1 2 3 4
tutorial : bash — Konsole
2locks : okular — Konsole
Programming Languages – O...
09:00:50

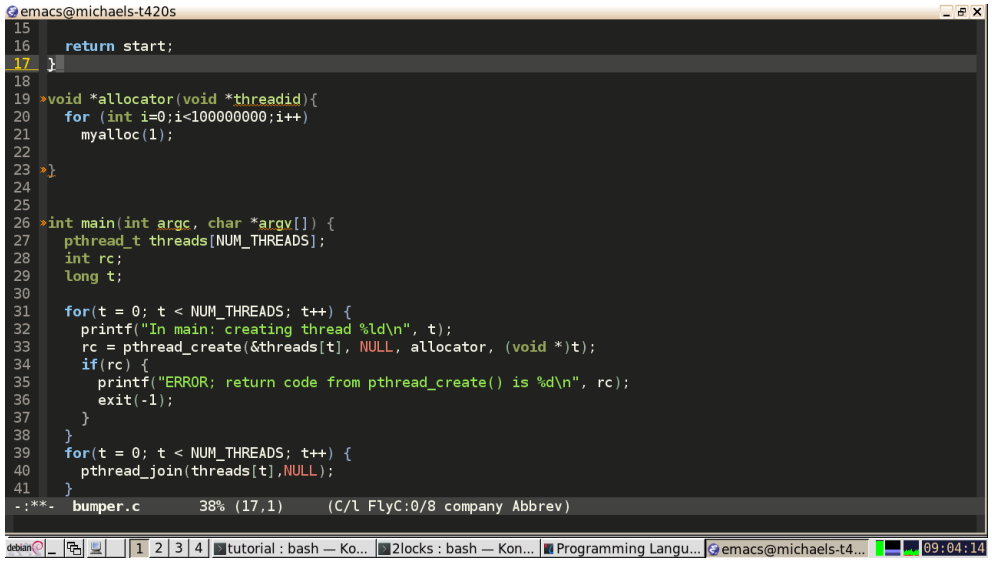

tutorial : bash — Konsole
Datei Bearbeiten Ansicht Lesezeichen Einstellungen Hilfe
tid #0 -> 1;
tid #1 -> 1;
tid #0 -> 1;
tid #1 -> 1;
tid #0 -> 1;
tid #0 -> 1;
tid #1 -> 1;
tid #0 ^C
petter@michaels-t420s:/home/petter/lehre/tutorial$ gcc -p
petter@michaels-t420s:/home/petter/lehre/tutorial$ ./dekk
In main: creating thread 0
In main: creating thread 1
This is thread #1!
This is thread #0!
^C
petter@michaels-t420s:/home/petter/lehre/tutorial$ Xlib:
[1]+ Fertig emacs dekker.c
petter@michaels-t420s:/home/petter/lehre/tutorial$ ls
bumper.c bumper.c~ dekker dekker.c dekker.c~ DQueue.
petter@michaels-t420s:/home/petter/lehre/tutorial$ emacs
[1] 2401
petter@michaels-t420s:/home/petter/lehre/tutorial$
tutorial : bash
Major opcode: 62 (X_CopyArea)
Resource id: 0x0
^Z
[1]+ Angehalten okular 2locks.pdf
petter@michaels-t420s:/home/petter/lehre/proglang/slides/
[1]+ okular 2locks.pdf &
petter@michaels-t420s:/home/petter/lehre/proglang/slides/
2locks : bash
emacs@michaels-t420s
1 2 3 4
tutorial : bash — Ko...
2locks : bash — Kon...
Programming Langu...
emacs@michaels-t4...
09:01:24

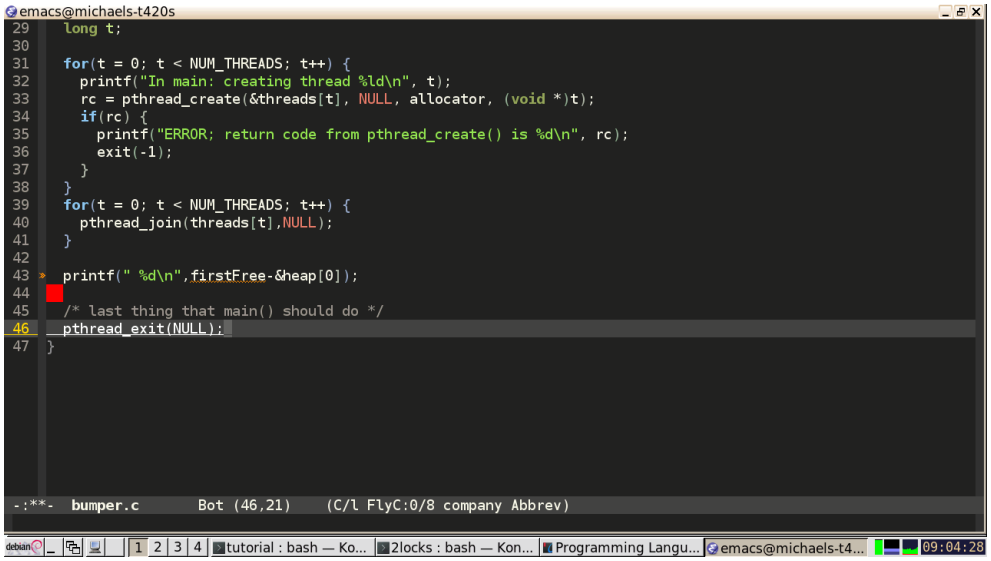

emacs@michaels-t420s
return start;
}
void *allocator(void *threadid){
for (int i=0;i<100000000;i++)
myalloc(1);
}
int main(int argc, char *argv[]) {
pthread_t threads[NUM_THREADS];
int rc;
long t;
for(t = 0; t < NUM_THREADS; t++) {
printf("In main: creating thread %ld\n", t);
rc = pthread_create(&threads[t], NULL, allocator, (void *)t);
if(rc) {
printf("ERROR; return code from pthread_create() is %d\n", rc);
exit(-1);
}
}
for(t = 0; t < NUM_THREADS; t++) {
pthread_join(threads[t],NULL);
}
-:**- bumper.c 38% (36,15) (C/l FlyC:0/8 company Abbrev)
1 2 3 4
tutorial : bash — Ko...
2locks : bash — Kon...
Programming Langu...
emacs@michaels-t4...
09:02:23

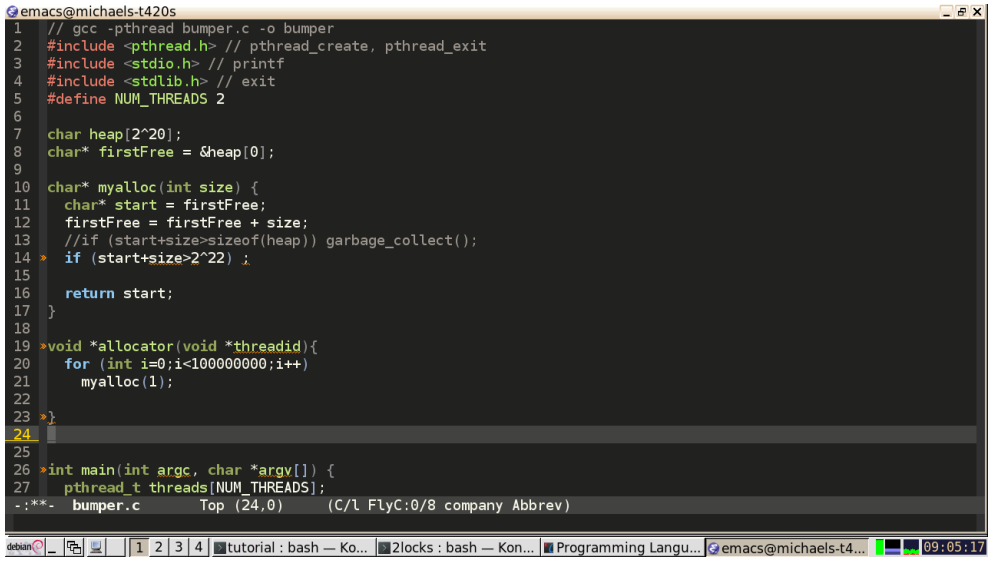

emacs@michaels-t420s
// gcc -pthread bumper.c -o bumper
#include <pthread.h> // pthread_create, pthread_exit
#include <stdio.h> // printf
#include <stdlib.h> // exit
#define NUM_THREADS 2
char heap[2^20];
char* firstFree = &heap[0];
char* myalloc(int size) {
char* start = firstFree;
firstFree = firstFree + size;
//if (start+siz >sizeof(heap)) garbage_collect();
if (start+size>2^22) ;
return start;
}
void *allocator(void *threadid){
for (int i=0;i<100000000;i++)
myalloc(1);
}
int main(int argc, char *argv[]) {
pthread_t threads[NUM_THREADS];
-:**- bumper.c Top (13,17) (C/l FlyC:0/8 company Abbrev)
1 2 3 4 tutorial : bash — Ko... 2locks : bash — Kon... Programming Langu... emacs@michaels-t4... 09:03:07

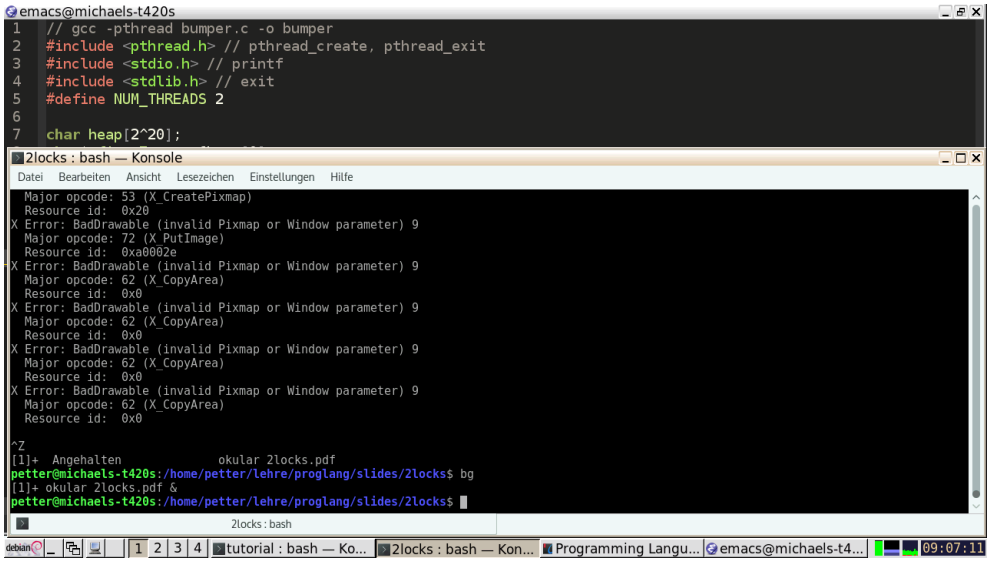

emacs@michaels-t420s
return start;
}
void *allocator(void *threadid){
for (int i=0;i<100000000;i++)
myalloc(1);
}
int main(int argc, char *argv[]) {
pthread_t threads[NUM_THREADS];
int rc;
long t;
for(t = 0; t < NUM_THREADS; t++) {
printf("In main: creating thread %ld\n", t);
rc = pthread_create(&threads[t], NULL, allocator, (void *)t);
if(rc) {
printf("ERROR; return code from pthread_create() is %d\n", rc);
exit(-1);
}
}
for(t = 0; t < NUM_THREADS; t++) {
pthread_join(threads[t],NULL);
}
-:**- bumper.c 38% (29,9) (C/l FlyC:0/8 company Abbrev)
1 2 3 4 tutorial : bash — Ko... 2locks : bash — Kon... Programming Langu... emacs@michaels-t4... 09:03:23

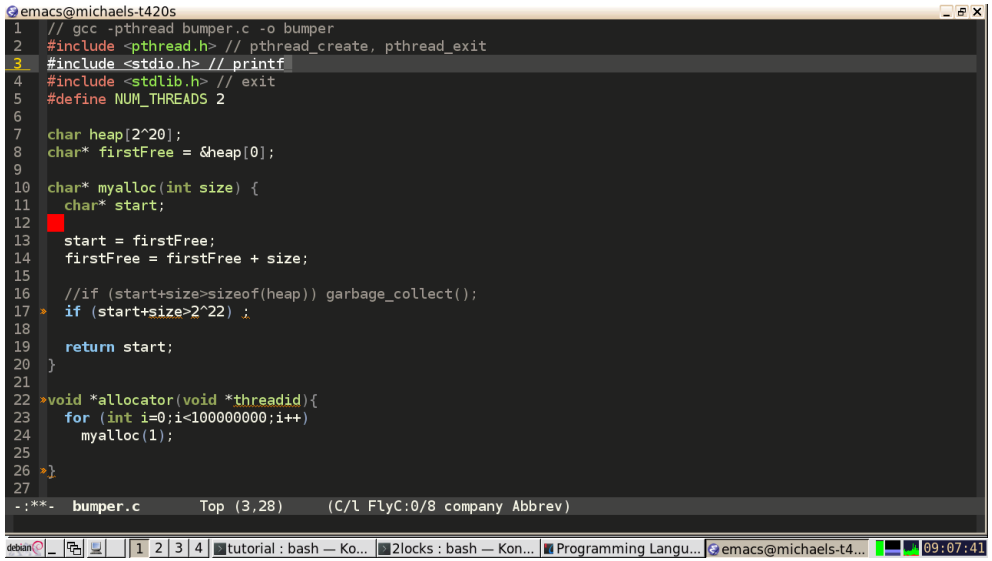

emacs@michaels-t420s
long t;
for(t = 0; t < NUM_THREADS; t++) {
printf("In main: creating thread %ld\n", t);
rc = pthread_create(&threads[t], NULL, allocator, (void *)t);
if(rc) {
printf("ERROR; return code from pthread_create() is %d\n", rc);
exit(-1);
}
}
for(t = 0; t < NUM_THREADS; t++) {
pthread_join(threads[t],NULL);
}
printf(" %d\n",firstFree-&heap[0]);
/* last thing that main() should do */
pthread_exit(NULL);
}
-:**- bumper.c Bot (47,1) (C/l FlyC:0/8 company Abbrev)
1 2 3 4 tutorial : bash — Ko... 2locks : bash — Kon... Programming Langu... emacs@michaels-t4... 09:03:48

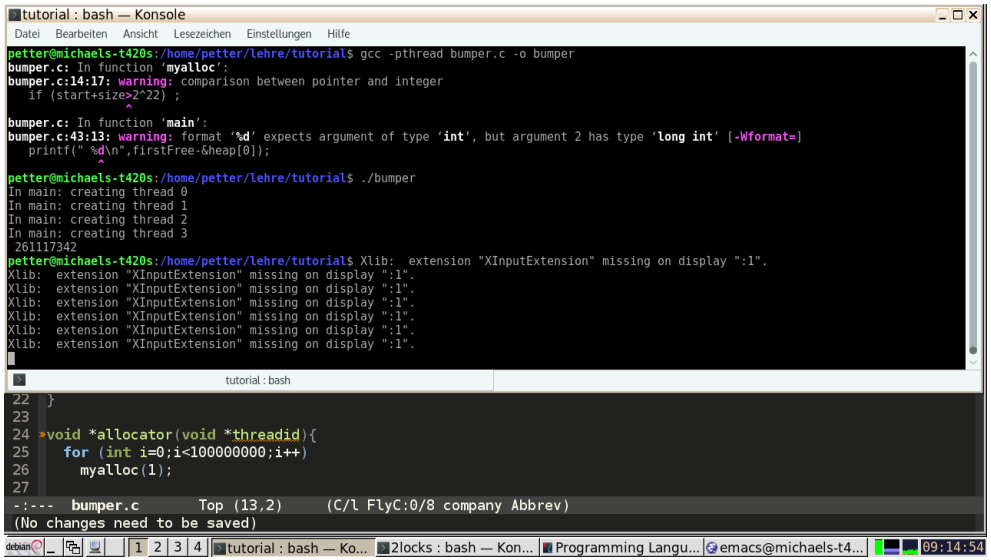

emacs@michaels-t420s
return start;
}
void *allocator(void *threadid){
for (int i=0;i<100000000;i++)
myalloc(1);
}
int main(int argc, char *argv[]) {
pthread_t threads[NUM_THREADS];
int rc;
long t;
for(t = 0; t < NUM_THREADS; t++) {
printf("In main: creating thread %ld\n", t);
rc = pthread_create(&threads[t], NULL, allocator, (void *)t);
if(rc) {
printf("ERROR; return code from pthread_create() is %d\n", rc);
exit(-1);
}
}
for(t = 0; t < NUM_THREADS; t++) {
pthread_join(threads[t],NULL);
}
-:**- bumper.c 38% (17,1) (C/l FlyC:0/8 company Abbrev)
1 2 3 4 tutorial : bash — Ko... 2locks : bash — Kon... Programming Langu... emacs@michaels-t4... 09:04:14

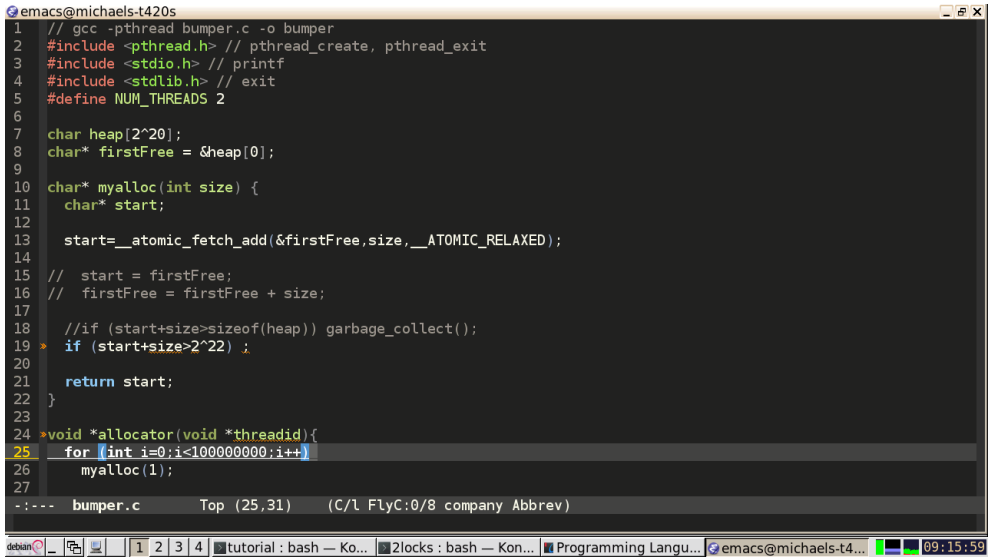

emacs@michaels-t420s
long t;
for(t = 0; t < NUM_THREADS; t++) {
printf("In main: creating thread %ld\n", t);
rc = pthread_create(&threads[t], NULL, allocator, (void *)t);
if(rc) {
printf("ERROR; return code from pthread_create() is %d\n", rc);
exit(-1);
}
}
for(t = 0; t < NUM_THREADS; t++) {
pthread_join(threads[t],NULL);
}
printf(" %d\n",firstFree-&heap[0]);
/* last thing that main() should do */
pthread_exit(NULL);
}
-:**- bumper.c Bot (46,21) (C/l FlyC:0/8 company Abbrev)
tutorial : bash — Ko...
2locks : bash — Kon...
Programming Langu...
emacs@michaels-t4...
09:04:28

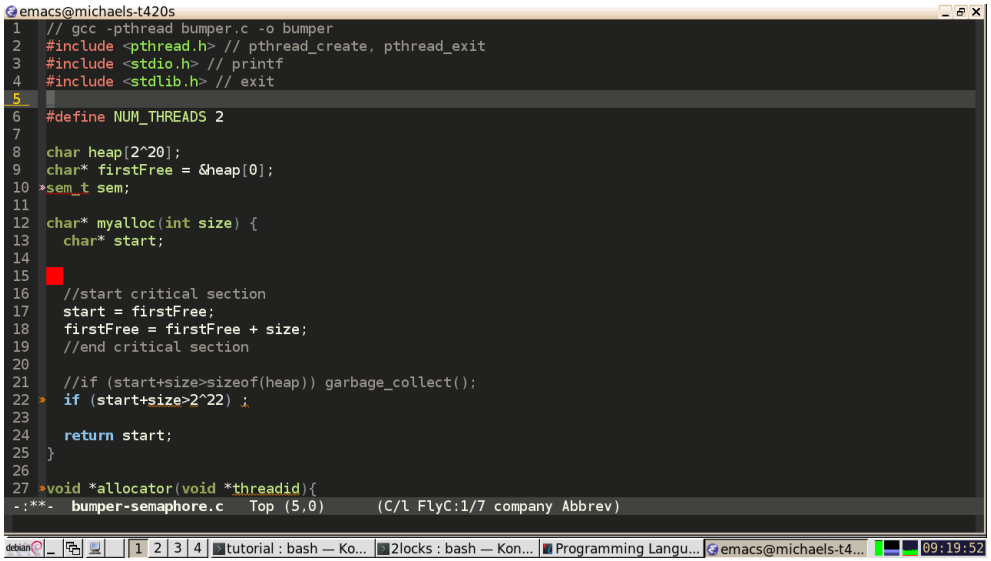

emacs@michaels-t420s
// gcc -pthread bumper.c -o bumper
#include <pthread.h> // pthread_create, pthread_exit
#include <stdio.h> // printf
#include <stdlib.h> // exit
#define NUM_THREADS 2
char heap[2^20];
char* firstFree = &heap[0];
char* myalloc(int size) {
char* start = firstFree;
firstFree = firstFree + size;
//if (start+size>sizeof(heap)) garbage_collect();
if (start+size>2^22) ;
return start;
}
void *allocator(void *threadid){
for (int i=0;i<100000000;i++)
myalloc(1);
}
int main(int argc, char *argv[]) {
pthread_t threads[NUM_THREADS];
-:**- bumper.c Top (24,0) (C/l FlyC:0/8 company Abbrev)
tutorial : bash — Ko...
2locks : bash — Kon...
Programming Langu...
emacs@michaels-t4...
09:05:17

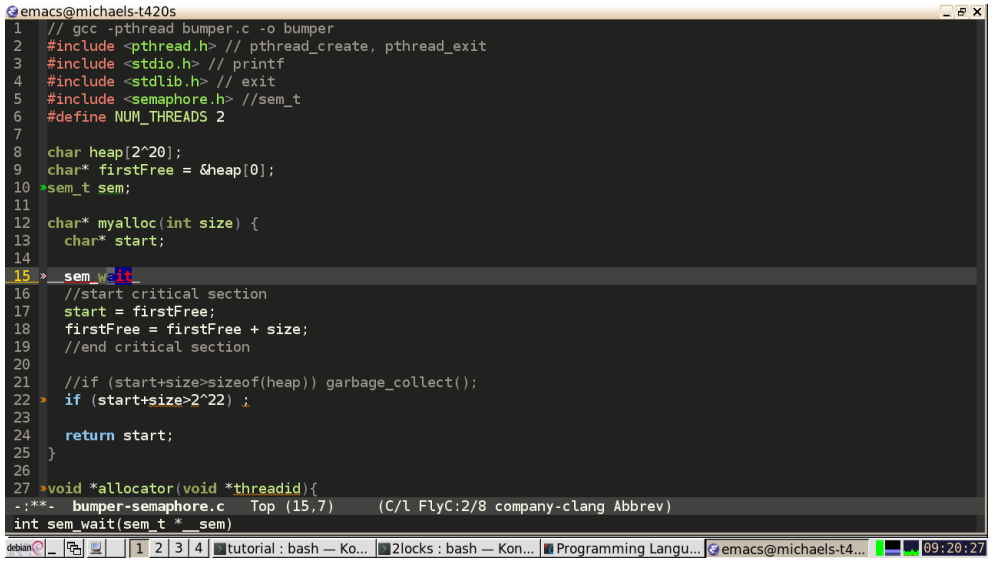

emacs@michaels-t420s
// gcc -pthread bumper.c -o bumper
#include <pthread.h> // pthread_create, pthread_exit
#include <stdio.h> // printf
#include <stdlib.h> // exit
#define NUM_THREADS 2
char heap[2^20];
2locks : bash — Konsole
Datei Bearbeiten Ansicht Lesezeichen Einstellungen Hilfe
Major opcode: 53 (X_CreatePixmap)
Resource id: 0x20
X Error: BadDrawable (invalid Pixmap or Window parameter) 9
Major opcode: 72 (X_PutImage)
Resource id: 0xa0002e
X Error: BadDrawable (invalid Pixmap or Window parameter) 9
Major opcode: 62 (X_CopyArea)
Resource id: 0x0
X Error: BadDrawable (invalid Pixmap or Window parameter) 9
Major opcode: 62 (X_CopyArea)
Resource id: 0x0
X Error: BadDrawable (invalid Pixmap or Window parameter) 9
Major opcode: 62 (X_CopyArea)
Resource id: 0x0
X Error: BadDrawable (invalid Pixmap or Window parameter) 9
Major opcode: 62 (X_CopyArea)
Resource id: 0x0
^Z
[1]+ Angehalten okular 2locks.pdf
petter@michaels-t420s:/home/petter/lehre/proglang/slides/2locks$ bg
[1]+ okular 2locks.pdf &
petter@michaels-t420s:/home/petter/lehre/proglang/slides/2locks$
2locks : bash
tutorial : bash — Ko...
2locks : bash — Kon...
Programming Langu...
emacs@michaels-t4...
09:07:11

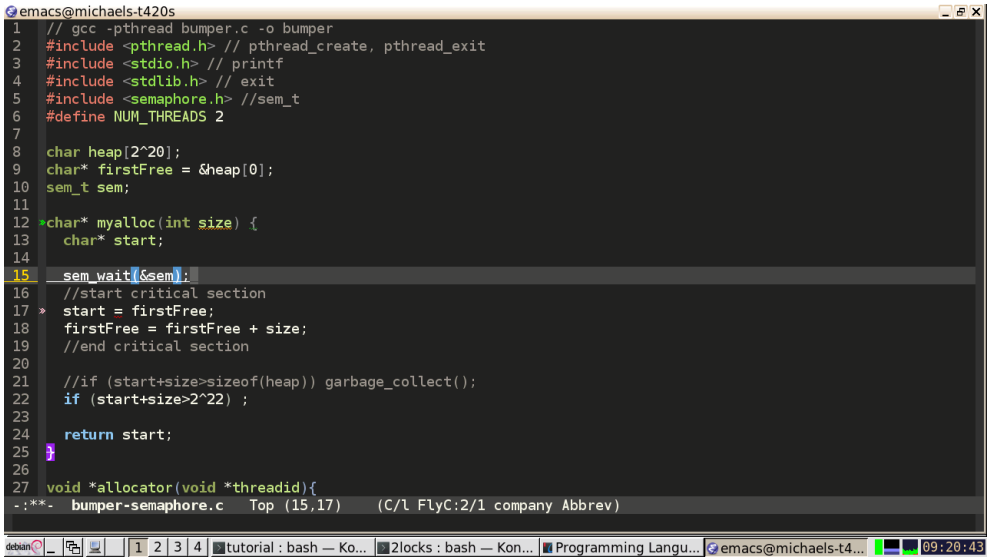

emacs@michaels-t420s
// gcc -pthread bumper.c -o bumper
#include <pthread.h> // pthread_create, pthread_exit
#include <stdio.h> // printf
#include <stdlib.h> // exit
#define NUM_THREADS 2
char heap[2^20];
char* firstFree = &heap[0];
char* myalloc(int size) {
char* start;
start = firstFree;
firstFree = firstFree + size;
//if (start+size>sizeof(heap)) garbage_collect();
if (start+size>2^22) ;
return start;
}
void *allocator(void *threadid){
for (int i=0;i<100000000;i++)
myalloc(1);
}
-:**- bumper.c Top (3,28) (C/l FlyC:0/8 company Abbrev)
tutorial : bash — Ko...
2locks : bash — Kon...
Programming Langu...
emacs@michaels-t4...
09:07:41

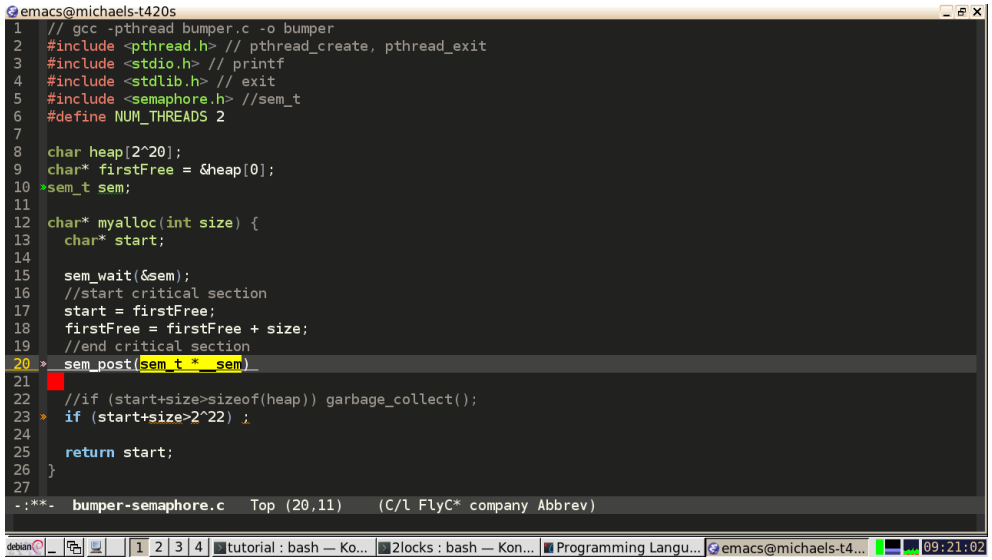

tutorial : bash — Konsole
Datei Bearbeiten Ansicht Lesezeichen Einstellungen Hilfe
petter@michaels-t420s:/home/petter/lehre/tutorial$ gcc -pthread bumper.c -o bumper
bumper.c: In function 'myalloc':
bumper.c:14:17: warning: comparison between pointer and integer
if (start+size>2^22) ;
bumper.c: In function 'main':
bumper.c:43:13: warning: format '%d' expects argument of type 'int', but argument 2 has type 'long int' [-Wformat=]
printf(" %d\n",firstFree-&heap[0]);
petter@michaels-t420s:/home/petter/lehre/tutorial$ ./bumper
In main: creating thread 0
In main: creating thread 1
In main: creating thread 2
In main: creating thread 3
261117342
petter@michaels-t420s:/home/petter/lehre/tutorial$ Xlib: extension "XInputExtension" missing on display ":1".
Xlib: extension "XInputExtension" missing on display ":1".
Xlib: extension "XInputExtension" missing on display ":1".
Xlib: extension "XInputExtension" missing on display ":1".
Xlib: extension "XInputExtension" missing on display ":1".
Xlib: extension "XInputExtension" missing on display ":1".
tutorial : bash
22 }
23
24 void *allocator(void *threadid){
25 for (int i=0;i<100000000;i++)
26 myalloc(1);
27
-:--- bumper.c Top (13,2) (C/l FlyC:0/8 company Abbrev)
(No changes need to be saved)
1 2 3 4 tutorial : bash — Ko... 2locks : bash — Kon... Programming Langu... emacs@michaels-t4... 09:14:54

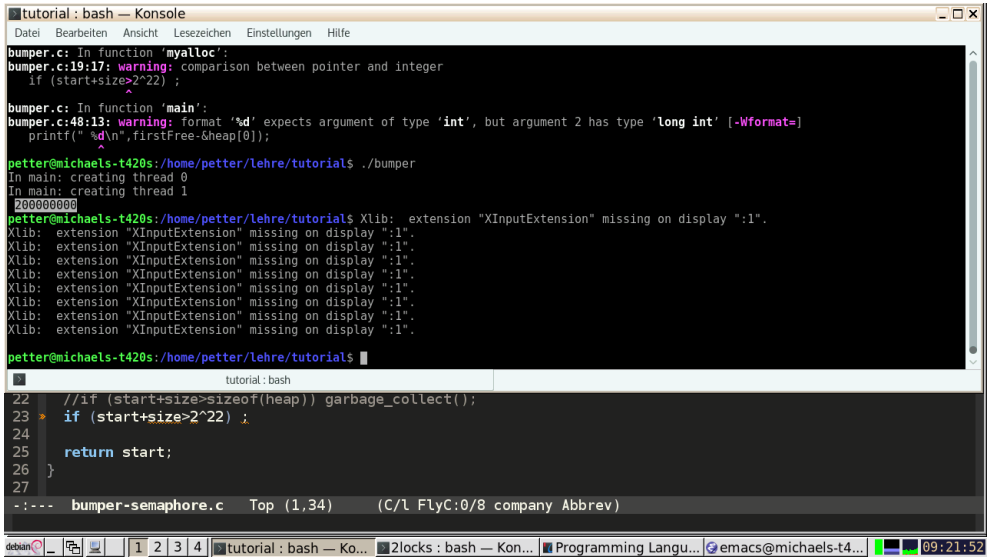

emacs@michaels-t420s
// gcc -pthread bumper.c -o bumper
#include <pthread.h> // pthread_create, pthread_exit
#include <stdio.h> // printf
#include <stdlib.h> // exit
#define NUM_THREADS 2
char heap[2^20];
char* firstFree = &heap[0];
char* myalloc(int size) {
char* start;
start=__atomic_fetch_add(&firstFree,size,__ATOMIC_RELAXED);
// start = firstFree;
// firstFree = firstFree + size;
//if (start+size>sizeof(heap)) garbage_collect();
if (start+size>2^22) ;
return start;
}
void *allocator(void *threadid){
for (int i=0;i<100000000;i++)
myalloc(1);
-:--- bumper.c Top (25,31) (C/l FlyC:0/8 company Abbrev)
1 2 3 4 tutorial : bash — Ko... 2locks : bash — Kon... Programming Langu... emacs@michaels-t4... 09:15:59

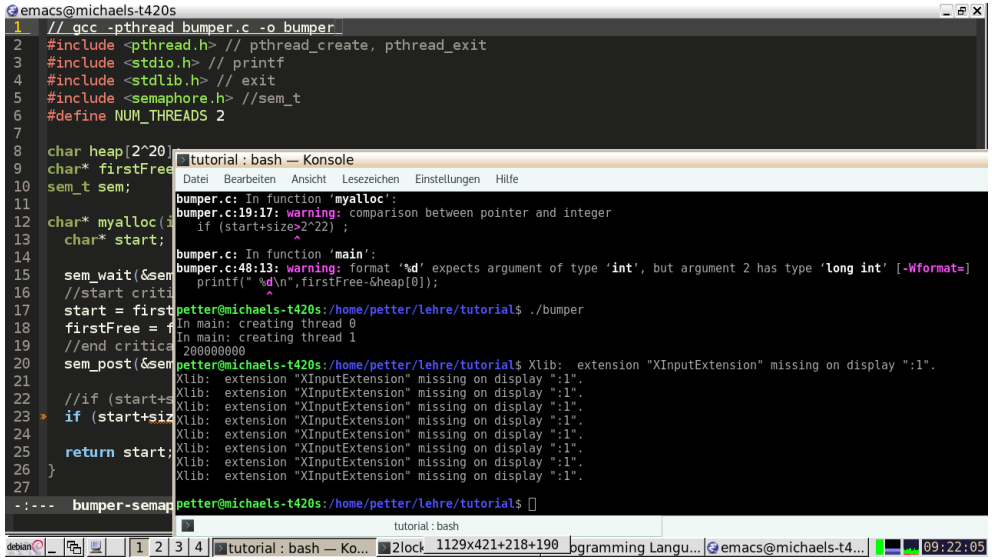

emacs@michaels-t420s
// gcc -pthread bumper.c -o bumper
#include <pthread.h> // pthread_create, pthread_exit
#include <stdio.h> // printf
#include <stdlib.h> // exit
#define NUM_THREADS 2
char heap[2^20];
char* firstFree = &heap[0];
sem_t sem;
char* myalloc(int size) {
char* start;
//start critical section
start = firstFree;
firstFree = firstFree + size;
//end critical section
//if (start+size>sizeof(heap)) garbage_collect();
if (start+size>2^22) ;
return start;
}
void *allocator(void *threadid){
-:**- bumper-semaphore.c Top (5,0) (C/l FlyC:1/7 company Abbrev)
1 2 3 4 tutorial : bash — Ko... 2locks : bash — Kon... Programming Langu... emacs@michaels-t4... 09:19:52

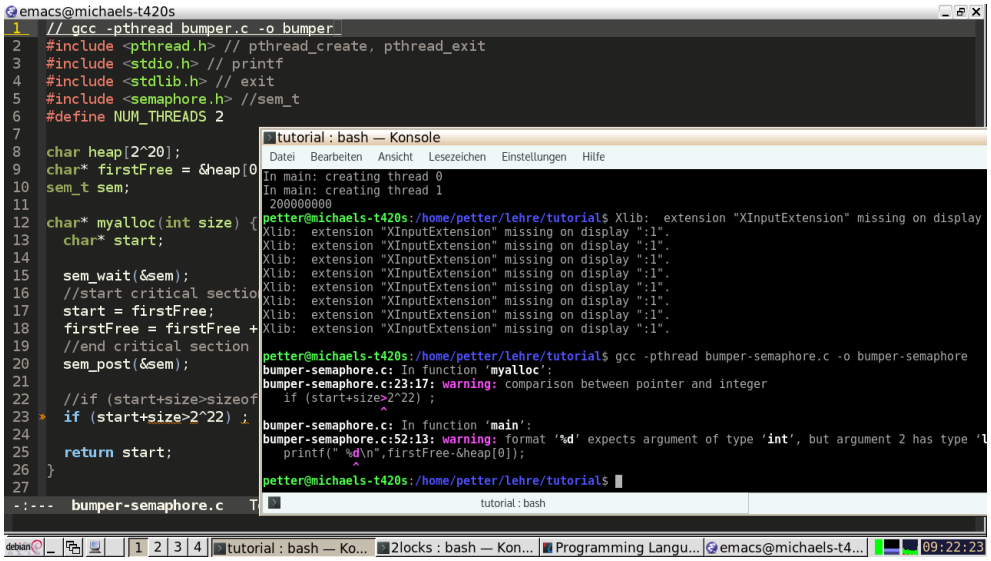

emacs@michaels-t420s
// gcc -pthread bumper.c -o bumper
#include <pthread.h> // pthread_create, pthread_exit
#include <stdio.h> // printf
#include <stdlib.h> // exit
#include <semaphore.h> //sem_t
#define NUM_THREADS 2
char heap[2^20];
char* firstFree = &heap[0];
sem_t sem;
char* myalloc(int size) {
char* start;
__sem_wait
//start critical section
start = firstFree;
firstFree = firstFree + size;
//end critical section
//if (start+size>sizeof(heap)) garbage_collect();
if (start+size>2^22) ;
return start;
}
void *allocator(void *threadid){
-:**- bumper-semaphore.c Top (15,7) (C/l FlyC:2/8 company-clang Abbrev)
int sem_wait(sem_t *__sem)
1 2 3 4 tutorial : bash — Ko... 2locks : bash — Kon... Programming Langu... emacs@michaels-t4... 09:20:27

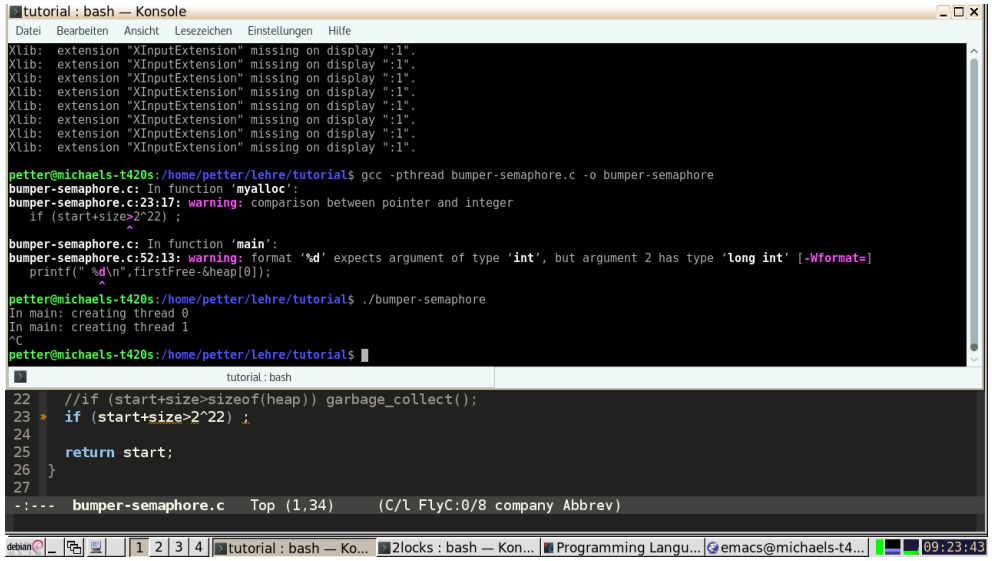

emacs@michaels-t420s
// gcc -pthread bumper.c -o bumper
#include <pthread.h> // pthread_create, pthread_exit
#include <stdio.h> // printf
#include <stdlib.h> // exit
#include <semaphore.h> //sem_t
#define NUM_THREADS 2
char heap[2^20];
char* firstFree = &heap[0];
sem_t sem;
char* myalloc(int size) {
char* start;
sem_wait(&sem);
//start critical section
start = firstFree;
firstFree = firstFree + size;
//end critical section
//if (start+size>sizeof(heap)) garbage_collect();
if (start+size>2^22) ;
return start;
}
void *allocator(void *threadid){
-:**- bumper-semaphore.c Top (15,17) (C/l FlyC:2/1 company Abbrev)
tutorial : bash — Ko...
2locks : bash — Kon...
Programming Langu...
emacs@michaels-t4...
09:20:43

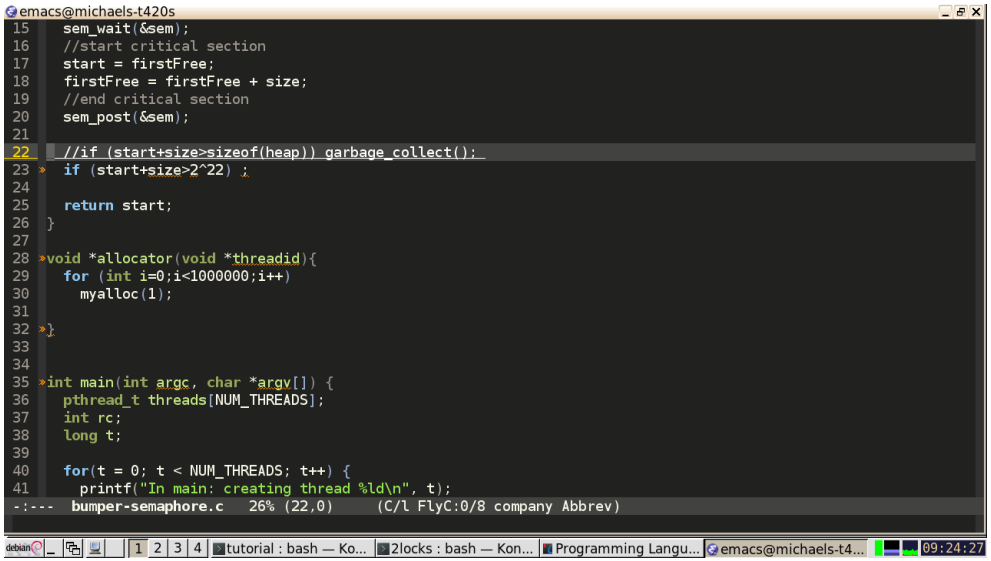

emacs@michaels-t420s
// gcc -pthread bumper.c -o bumper
#include <pthread.h> // pthread_create, pthread_exit
#include <stdio.h> // printf
#include <stdlib.h> // exit
#include <semaphore.h> //sem_t
#define NUM_THREADS 2
char heap[2^20];
char* firstFree = &heap[0];
sem_t sem;
char* myalloc(int size) {
char* start;
sem_wait(&sem);
//start critical section
start = firstFree;
firstFree = firstFree + size;
//end critical section
sem_post(sem_t * __sem)
//if (start+size>sizeof(heap)) garbage_collect();
if (start+size>2^22) ;
return start;
}
-:**- bumper-semaphore.c Top (20,11) (C/l FlyC* company Abbrev)
tutorial : bash — Ko...
2locks : bash — Kon...
Programming Langu...
emacs@michaels-t4...
09:21:02

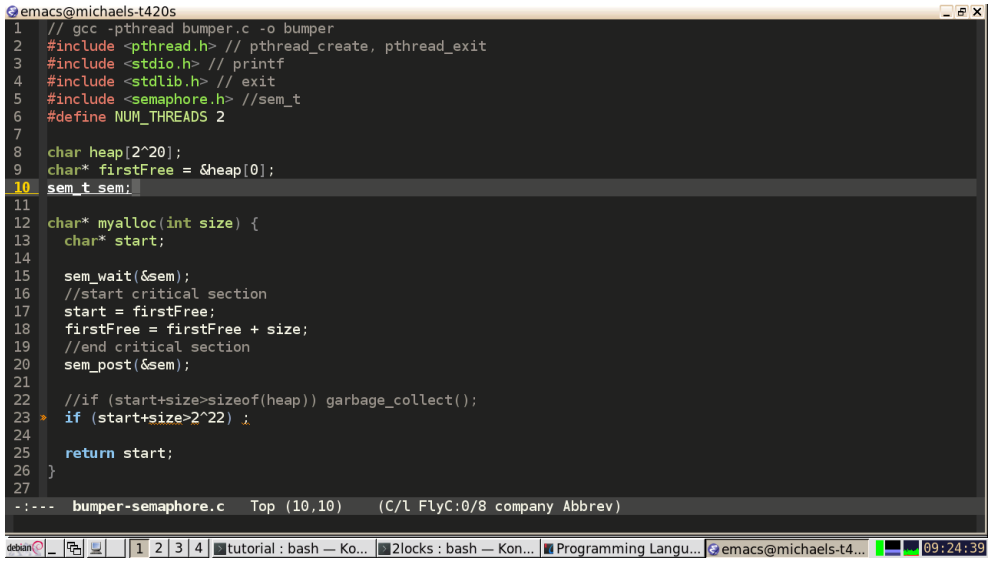

tutorial : bash — Konsole
Datei Bearbeiten Ansicht Lesezeichen Einstellungen Hilfe
bumper.c: In function 'myalloc':
bumper.c:19:17: warning: comparison between pointer and integer
if (start+size>2^22) ;
bumper.c: In function 'main':
bumper.c:48:13: warning: format '%d' expects argument of type 'int', but argument 2 has type 'long int' [-Wformat=]
printf(" %d\n",firstFree-&heap[0]);
petter@michaels-t420s:/home/petter/lehre/tutorial$ ./bumper
In main: creating thread 0
In main: creating thread 1
200000000
petter@michaels-t420s:/home/petter/lehre/tutorial$ Xlib: extension "XInputExtension" missing on display ":1".
tutorial : bash
-:--- bumper-semaphore.c Top (1,34) (C/l FlyC:0/8 company Abbrev)
tutorial : bash — Ko...
2locks : bash — Kon...
Programming Langu...
emacs@michaels-t4...
09:21:52

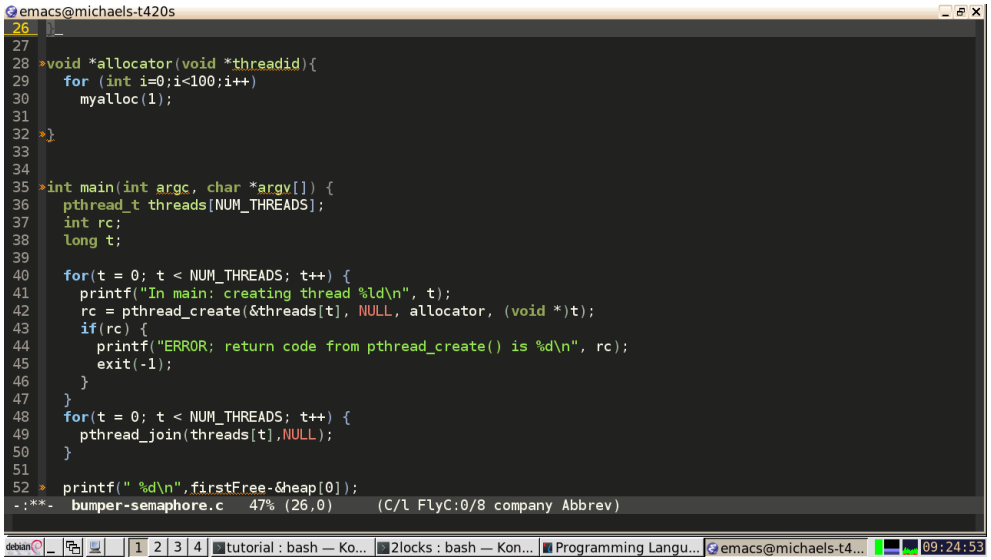

emacs@michaels-t420s
// gcc -pthread bumper.c -o bumper
#include <pthread.h> // pthread_create, pthread_exit
#include <stdio.h> // printf
#include <stdlib.h> // exit
#include <semaphore.h> //sem_t
#define NUM_THREADS 2
tutorial : bash — Konsole
Datei Bearbeiten Ansicht Lesezeichen Einstellungen Hilfe
bumper.c: In function 'myalloc':
bumper.c:19:17: warning: comparison between pointer and integer
if (start+size>2^22) ;
bumper.c: In function 'main':
bumper.c:48:13: warning: format '%d' expects argument of type 'int', but argument 2 has type 'long int' [-Wformat=]
printf(" %d\n",firstFree-&heap[0]);
petter@michaels-t420s:/home/petter/lehre/tutorial$ ./bumper
In main: creating thread 0
In main: creating thread 1
200000000
petter@michaels-t420s:/home/petter/lehre/tutorial$ Xlib: extension "XInputExtension" missing on display ":1".
tutorial : bash
1129x421+218+190
tutorial : bash — Ko...
emacs@michaels-t4...
09:22:05

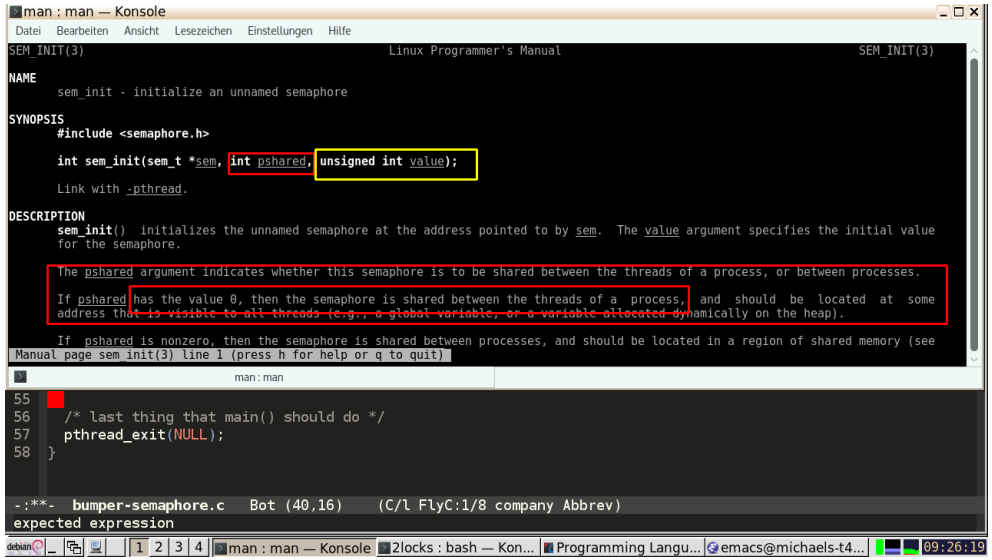

emacs@michaels-t420s
// gcc -pthread bumper.c -o bumper
#include <pthread.h> // pthread_create, pthread_exit
#include <stdio.h> // printf
#include <stdlib.h> // exit
#include <semaphore.h> //sem_t
#define NUM_THREADS 2
char heap[2^20];
char* firstFree = &heap[0
sem_t sem;
char* myalloc(int size) {
char* start;
sem_wait(&sem);
//start critical sectio
start = firstFree;
firstFree = firstFree +
//end critical section
sem_post(&sem);
//if (start+size>sizeof
if (start+size>2^22) ;
return start;
}
-:--- bumper-semaphore.c
tutorial : bash — Konsole
Datei Bearbeiten Ansicht Lesezeichen Einstellungen Hilfe
In main: creating thread 0
In main: creating thread 1
200000000
petter@michaels-t420s:/home/petter/lehre/tutorial$ Xlib: extension "XInputExtension" missing on display
Xlib: extension "XInputExtension" missing on display ":1".
petter@michaels-t420s:/home/petter/lehre/tutorial$ gcc -pthread bumper-semaphore.c -o bumper-semaphore
bumper-semaphore.c: In function 'myalloc':
bumper-semaphore.c:23:17: warning: comparison between pointer and integer
if (start+size>2^22) ;
bumper-semaphore.c: In function 'main':
bumper-semaphore.c:52:13: warning: format '%d' expects argument of type 'int', but argument 2 has type 'l
printf(" %d\n",firstFree-&heap[0]);
petter@michaels-t420s:/home/petter/lehre/tutorial$
tutorial : bash
debian 1 2 3 4 tutorial : bash — Ko... 2locks : bash — Kon... Programming Langu... emacs@michaels-t4... 09:22:23

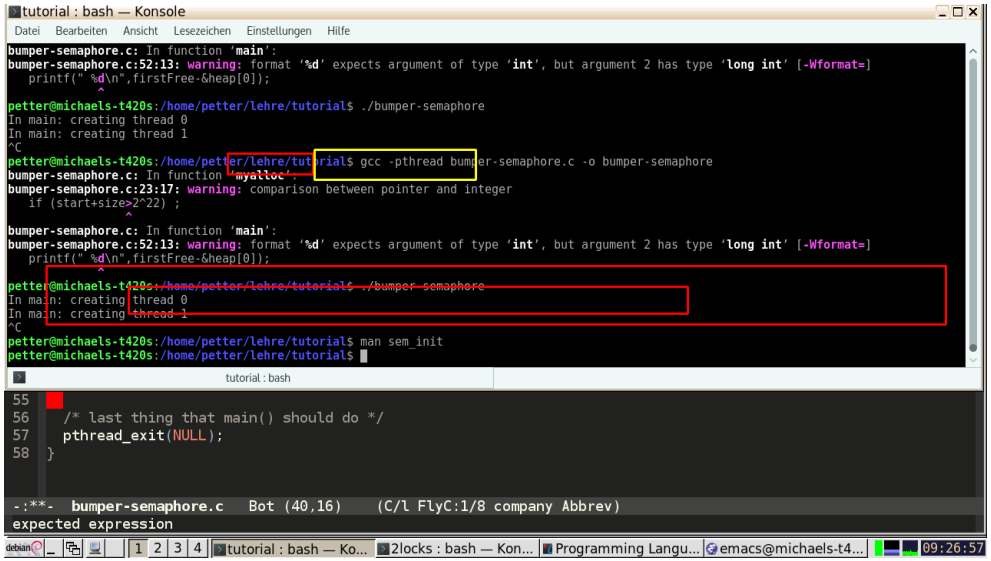

tutorial : bash — Konsole
Datei Bearbeiten Ansicht Lesezeichen Einstellungen Hilfe
Xlib: extension "XInputExtension" missing on display ":1".
petter@michaels-t420s:/home/petter/lehre/tutorial$ gcc -pthread bumper-semaphore.c -o bumper-semaphore
bumper-semaphore.c: In function 'myalloc':
bumper-semaphore.c:23:17: warning: comparison between pointer and integer
if (start+size>2^22) ;
bumper-semaphore.c: In function 'main':
bumper-semaphore.c:52:13: warning: format '%d' expects argument of type 'int', but argument 2 has type 'long int' [-Wformat=]
printf(" %d\n",firstFree-&heap[0]);
petter@michaels-t420s:/home/petter/lehre/tutorial$ ./bumper-semaphore
In main: creating thread 0
In main: creating thread 1
^C
petter@michaels-t420s:/home/petter/lehre/tutorial$
tutorial : bash
//if (start+size>sizeof(heap)) garbage_collect();
if (start+size>2^22) ;
return start;
}
-:--- bumper-semaphore.c Top (1,34) (C/l FlyC:0/8 company Abbrev)
debian 1 2 3 4 tutorial : bash — Ko... 2locks : bash — Kon... Programming Langu... emacs@michaels-t4... 09:23:43

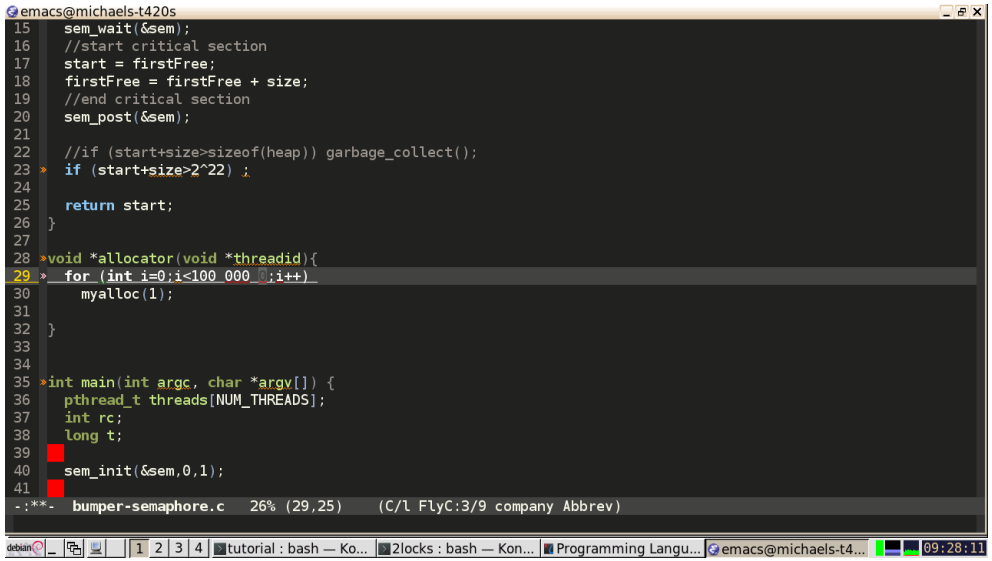

emacs@michaels-t420s
sem_wait(&sem);
//start critical section
start = firstFree;
firstFree = firstFree + size;
//end critical section
sem_post(&sem);
//if (start+size>sizeof(heap)) garbage_collect();
if (start+size>2^22) ;
return start;
}
void *allocator(void *threadid){
for (int i=0;i<1000000;i++)
myalloc(1);
}
int main(int argc, char *argv[]) {
pthread_t threads[NUM_THREADS];
int rc;
long t;
for(t = 0; t < NUM_THREADS; t++) {
printf("In main: creating thread %ld\n", t);
-:--- bumper-semaphore.c 26% (22,0) (C/l FlyC:0/8 company Abbrev)
debian 1 2 3 4 tutorial : bash — Ko... 2locks : bash — Kon... Programming Langu... emacs@michaels-t4... 09:24:27

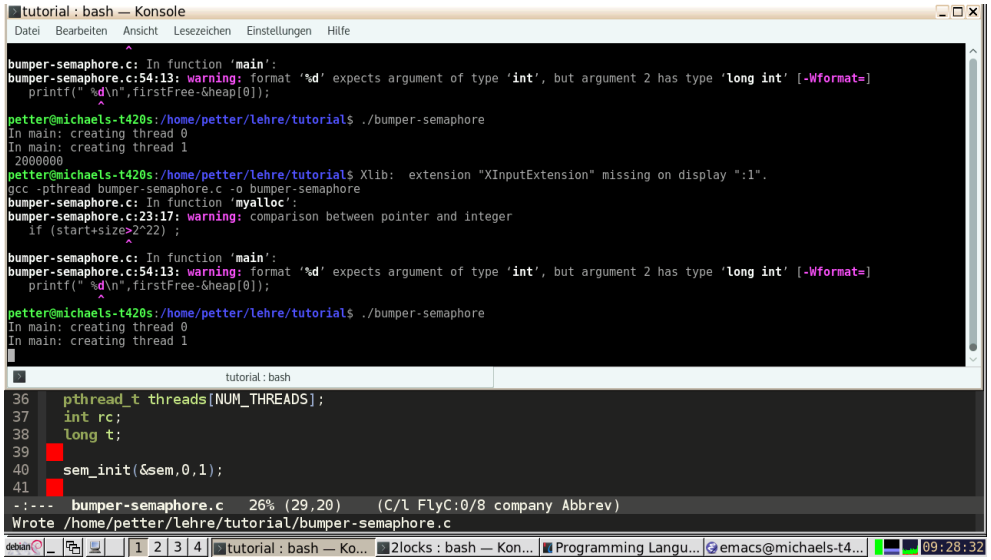

emacs@michaels-t420s
// gcc -pthread bumper.c -o bumper
#include <pthread.h> // pthread_create, pthread_exit
#include <stdio.h> // printf
#include <stdlib.h> // exit
#include <semaphore.h> //sem_t
#define NUM_THREADS 2
char heap[2^20];
char* firstFree = &heap[0];
sem_t sem;
char* myalloc(int size) {
char* start;
sem_wait(&sem);
//start critical section
start = firstFree;
firstFree = firstFree + size;
//end critical section
sem_post(&sem);
//if (start+size>sizeof(heap)) garbage_collect();
if (start+size>2^22) ;
return start;
}
-:--- bumper-semaphore.c Top (10,10) (C/l FlyC:0/8 company Abbrev)
debian 1 2 3 4 tutorial : bash — Ko... 2locks : bash — Kon... Programming Langu... emacs@michaels-t4... 09:24:39

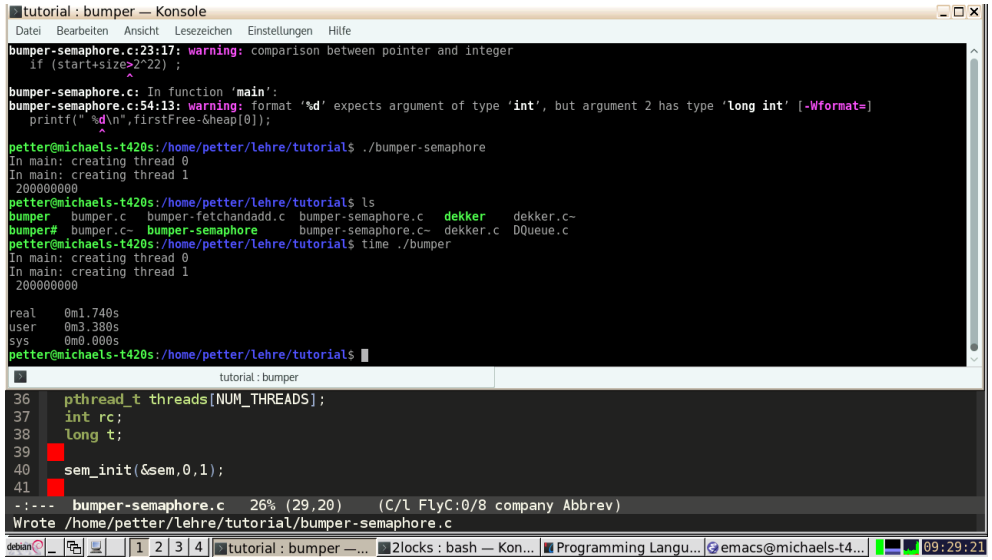

emacs@michaels-t420s
26
27
28 void *allocator(void *threadid){
29 for (int i=0;i<100;i++)
30 myalloc(1);
31
32 }
33
34
35 int main(int argc, char *argv[]) {
36 pthread_t threads[NUM_THREADS];
37 int rc;
38 long t;
39
40 for(t = 0; t < NUM_THREADS; t++) {
41 printf("In main: creating thread %ld\n", t);
42 rc = pthread_create(&threads[t], NULL, allocator, (void *)t);
43 if(rc) {
44 printf("ERROR; return code from pthread_create() is %d\n", rc);
45 exit(-1);
46 }
47 }
48 for(t = 0; t < NUM_THREADS; t++) {
49 pthread_join(threads[t],NULL);
50 }
51
52 printf(" %d\n",firstFree-&heap[0]);
-:**- bumper-semaphore.c 47% (26,0) (C/l FlyC:0/8 company Abbrev)
1 2 3 4
tutorial : bash — Ko...
2locks : bash — Kon...
Programming Langu...
emacs@michaels-t4...
09:24:53

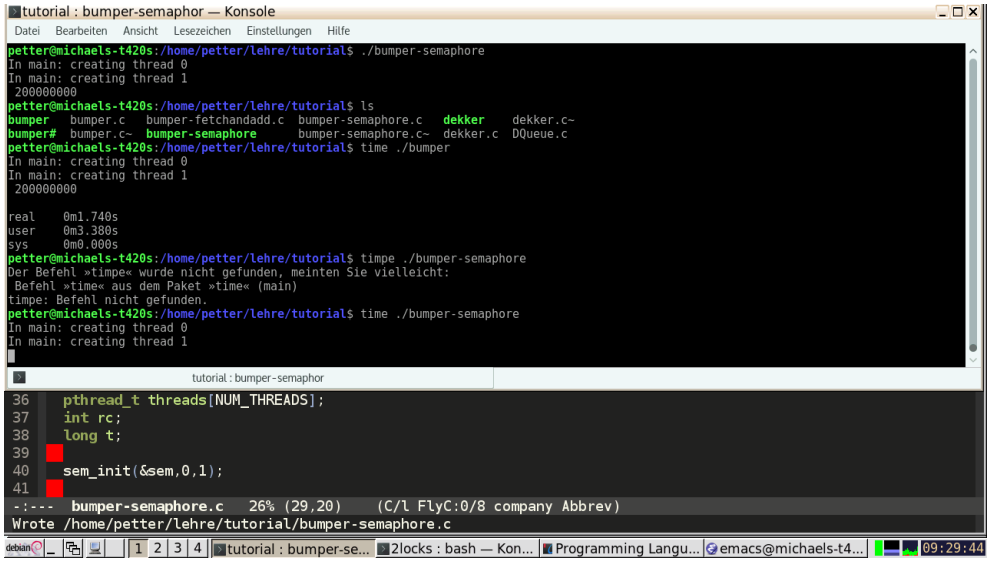

man : man — Konsole
Datei Bearbeiten Ansicht Lesezeichen Einstellungen Hilfe
SEM_INIT(3) Linux Programmer's Manual SEM_INIT(3)
NAME
sem_init - initialize an unnamed semaphore
SYNOPSIS
#include <semaphore.h>
int sem_init(sem_t *sem, int pshared, unsigned int value);
Link with -pthread.
DESCRIPTION
sem_init() initializes the unnamed semaphore at the address pointed to by sem. The value argument specifies the initial value for the semaphore.
The pshared argument indicates whether this semaphore is to be shared between the threads of a process, or between processes.
If pshared has the value 0, then the semaphore is shared between the threads of a process, and should be located at some address that is visible to all threads (e.g., a global variable, or a variable allocated dynamically on the heap).
If pshared is nonzero, then the semaphore is shared between processes, and should be located in a region of shared memory (see
Manual page sem_init(3) line 1 (press h for help or q to quit)
man : man
55
56 /* last thing that main() should do */
57 pthread_exit(NULL);
58 }
-:**- bumper-semaphore.c Bot (40,16) (C/l FlyC:1/8 company Abbrev)
expected expression
1 2 3 4
tutorial : bash — Ko...
2locks : bash — Kon...
Programming Langu...
emacs@michaels-t4...
09:26:19

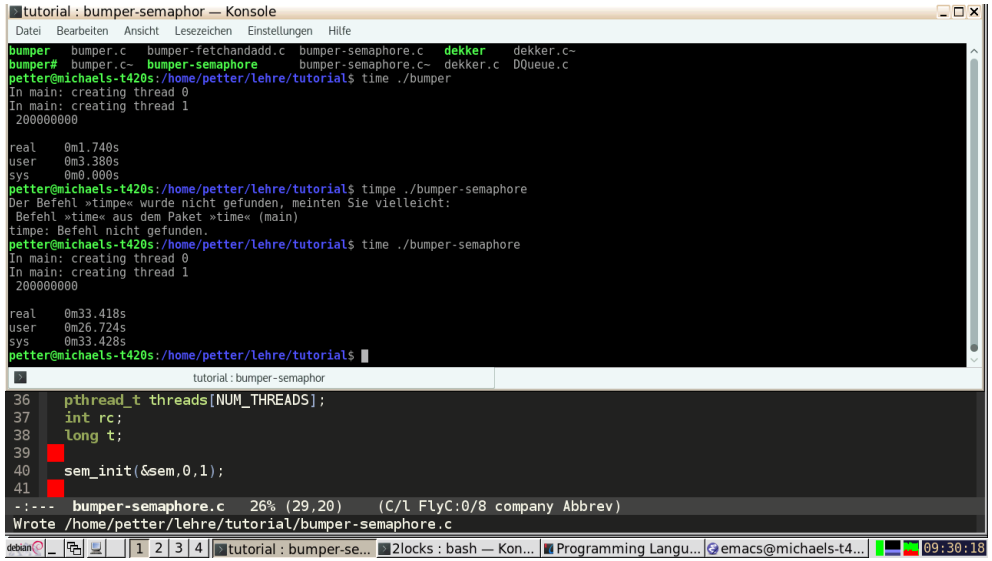

tutorial : bash — Konsole
Datei Bearbeiten Ansicht Lesezeichen Einstellungen Hilfe
bumper-semaphore.c: In function 'main':
bumper-semaphore.c:52:13: warning: format '%d' expects argument of type 'int', but argument 2 has type 'long int' [-Wformat=]
printf(" %d\n",firstFree-&heap[0]);
^
petter@michaels-t420s:/home/petter/lehre/tutorial$ ./bumper-semaphore
In main: creating thread 0
In main: creating thread 1
^C
petter@michaels-t420s:/home/petter/lehre/tutorial$ gcc -pthread bumper-semaphore.c -o bumper-semaphore
bumper-semaphore.c: In function 'myalloc':
bumper-semaphore.c:23:17: warning: comparison between pointer and integer
if (start+size>2^22) ;
^
bumper-semaphore.c: In function 'main':
bumper-semaphore.c:52:13: warning: format '%d' expects argument of type 'int', but argument 2 has type 'long int' [-Wformat=]
printf(" %d\n",firstFree-&heap[0]);
^
petter@michaels-t420s:/home/petter/lehre/tutorial$ ./bumper-semaphore
In main: creating thread 0
In main: creating thread 1
^C
petter@michaels-t420s:/home/petter/lehre/tutorial$ man sem_init
petter@michaels-t420s:/home/petter/lehre/tutorial$
tutorial : bash
55
56 /* last thing that main() should do */
57 pthread_exit(NULL);
58 }
-:**- bumper-semaphore.c Bot (40,16) (C/l FlyC:1/8 company Abbrev)
expected expression
1 2 3 4
tutorial : bash — Ko...
2locks : bash — Kon...
Programming Langu...
emacs@michaels-t4...
09:26:57

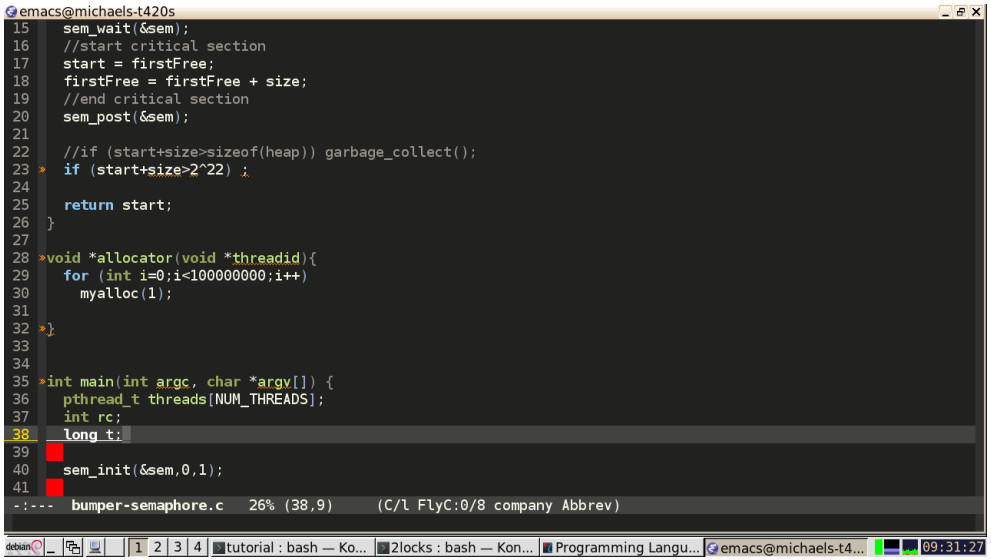

emacs@michaels-t420s
15 sem_wait(&sem);
16 //start critical section
17 start = firstFree;
18 firstFree = firstFree + size;
19 //end critical section
20 sem_post(&sem);
21
22 //if (start+size>sizeof(heap)) garbage_collect();
23 if (start+size>2^22) ;
24
25 return start;
26 }
27
28 void *allocator(void *threadid){
29 for (int i=0;i<100_000_000;i++)
30 myalloc(1);
31
32 }
33
34
35 int main(int argc, char *argv[]) {
36 pthread_t threads[NUM_THREADS];
37 int rc;
38 long t;
39
40 sem_init(&sem,0,1);
41
-:**- bumper-semaphore.c 26% (29,25) (C/l FlyC:3/9 company Abbrev)
1 2 3 4
tutorial : bash — Ko...
2locks : bash — Kon...
Programming Langu...
emacs@michaels-t4...
09:28:11

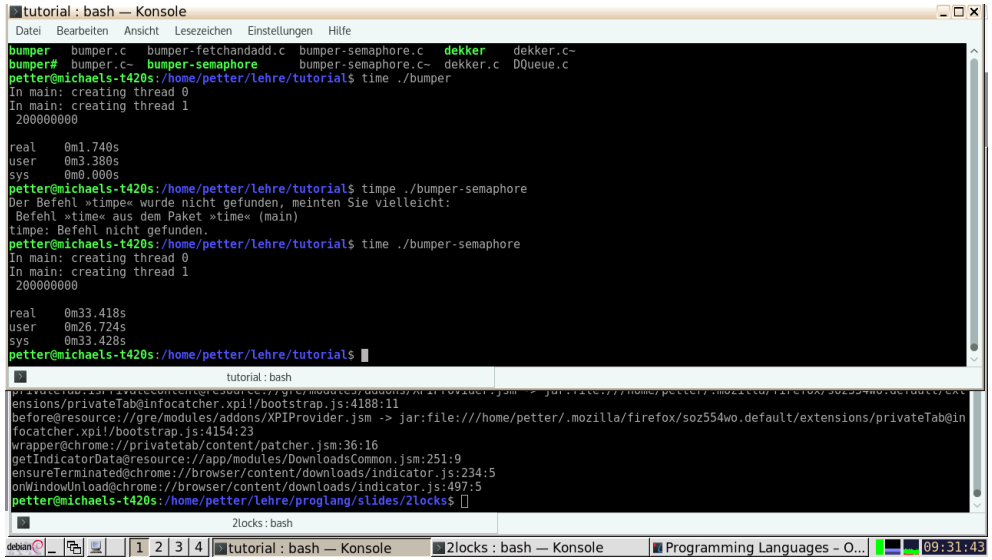

tutorial : bash — Konsole
Datei Bearbeiten Ansicht Lesezeichen Einstellungen Hilfe
bumper-semaphore.c: In function 'main':
bumper-semaphore.c:54:13: warning: format '%d' expects argument of type 'int', but argument 2 has type 'long int' [-Wformat=]
printf(" %d\n",firstFree-&heap[0]);
petter@michaels-t420s:/home/petter/lehre/tutorial$ ./bumper-semaphore
In main: creating thread 0
In main: creating thread 1
2000000
petter@michaels-t420s:/home/petter/lehre/tutorial$ Xlib: extension "XInputExtension" missing on display ":1".
gcc -pthread bumper-semaphore.c -o bumper-semaphore
bumper-semaphore.c: In function 'myalloc':
bumper-semaphore.c:23:17: warning: comparison between pointer and integer
if (start+size>2^22) ;
bumper-semaphore.c: In function 'main':
bumper-semaphore.c:54:13: warning: format '%d' expects argument of type 'int', but argument 2 has type 'long int' [-Wformat=]
printf(" %d\n",firstFree-&heap[0]);
petter@michaels-t420s:/home/petter/lehre/tutorial$ ./bumper-semaphore
In main: creating thread 0
In main: creating thread 1
tutorial : bash
pthread_t threads[NUM_THREADS];
int rc;
long t;
sem_init(&sem,0,1);
-:--- bumper-semaphore.c 26% (29,20) (C/l FlyC:0/8 company Abbrev)
Wrote /home/petter/lehre/tutorial/bumper-semaphore.c
tutorial : bash — Ko...
2locks : bash — Kon...
Programming Langu...
emacs@michaels-t4...
09:28:32

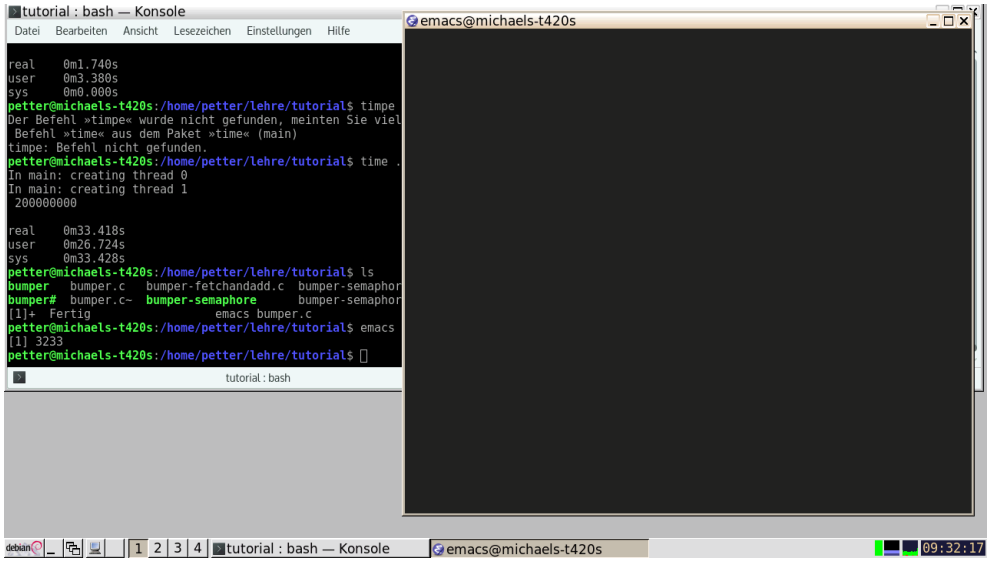

tutorial : bumper — Konsole
Datei Bearbeiten Ansicht Lesezeichen Einstellungen Hilfe
bumper-semaphore.c:23:17: warning: comparison between pointer and integer
if (start+size>2^22) ;
bumper-semaphore.c: In function 'main':
bumper-semaphore.c:54:13: warning: format '%d' expects argument of type 'int', but argument 2 has type 'long int' [-Wformat=]
printf(" %d\n",firstFree-&heap[0]);
petter@michaels-t420s:/home/petter/lehre/tutorial$ ./bumper-semaphore
In main: creating thread 0
In main: creating thread 1
200000000
petter@michaels-t420s:/home/petter/lehre/tutorial$ ls
bumper bumper.c bumper-fetchandadd.c bumper-semaphore.c dekker dekker.c~
bumper# bumper.c~ bumper-semaphore bumper-semaphore.c~ dekker.c DQueue.c
petter@michaels-t420s:/home/petter/lehre/tutorial$ time ./bumper
In main: creating thread 0
In main: creating thread 1
200000000
real 0m1.740s
user 0m3.380s
sys 0m0.000s
petter@michaels-t420s:/home/petter/lehre/tutorial$
tutorial : bumper
pthread_t threads[NUM_THREADS];
int rc;
long t;
sem_init(&sem,0,1);
-:--- bumper-semaphore.c 26% (29,20) (C/l FlyC:0/8 company Abbrev)
Wrote /home/petter/lehre/tutorial/bumper-semaphore.c
tutorial : bumper —...
2locks : bash — Kon...
Programming Langu...
emacs@michaels-t4...
09:29:21

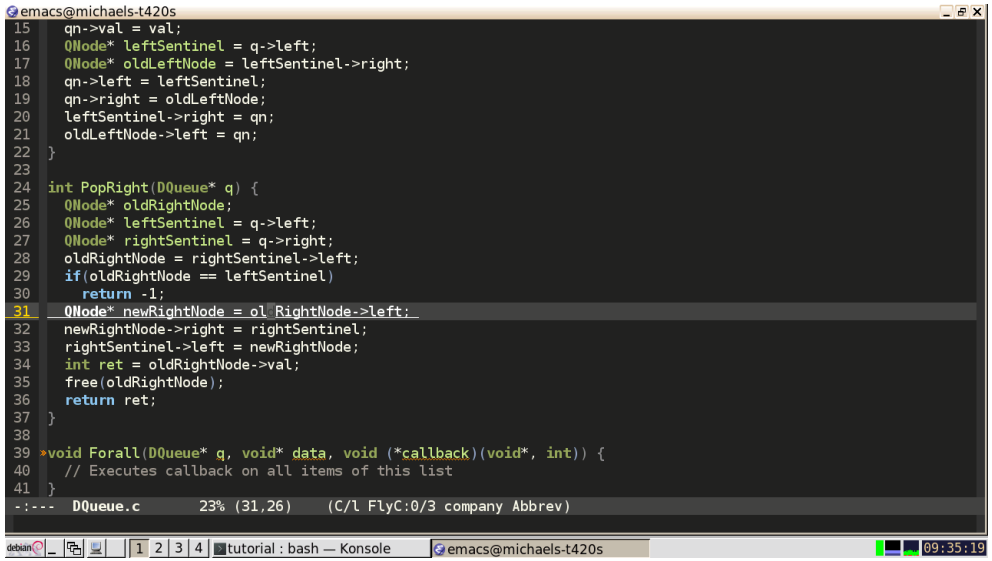

tutorial : bumper-semaphor — Konsole
Datei Bearbeiten Ansicht Lesezeichen Einstellungen Hilfe
petter@michaels-t420s:/home/petter/lehre/tutorial$ ./bumper-semaphore
In main: creating thread 0
In main: creating thread 1
200000000
petter@michaels-t420s:/home/petter/lehre/tutorial$ ls
bumper bumper.c bumper-fetchandadd.c bumper-semaphore.c dekker dekker.c~
bumper# bumper.c~ bumper-semaphore bumper-semaphore.c~ dekker.c DQueue.c
petter@michaels-t420s:/home/petter/lehre/tutorial$ time ./bumper
In main: creating thread 0
In main: creating thread 1
200000000
real 0m1.740s
user 0m3.380s
sys 0m0.000s
petter@michaels-t420s:/home/petter/lehre/tutorial$ timpe ./bumper-semaphore
Der Befehl »timpe« wurde nicht gefunden, meinten Sie vielleicht:
Befehl »time« aus dem Paket »time« (main)
timpe: Befehl nicht gefunden.
petter@michaels-t420s:/home/petter/lehre/tutorial$ time ./bumper-semaphore
In main: creating thread 0
In main: creating thread 1
tutorial : bumper-semaphor
pthread_t threads[NUM_THREADS];
int rc;
long t;
sem_init(&sem,0,1);
-:--- bumper-semaphore.c 26% (29,20) (C/l FlyC:0/8 company Abbrev)
Wrote /home/petter/lehre/tutorial/bumper-semaphore.c
tutorial : bumper-se...
2locks : bash — Kon...
Programming Langu...
emacs@michaels-t4...
09:29:44

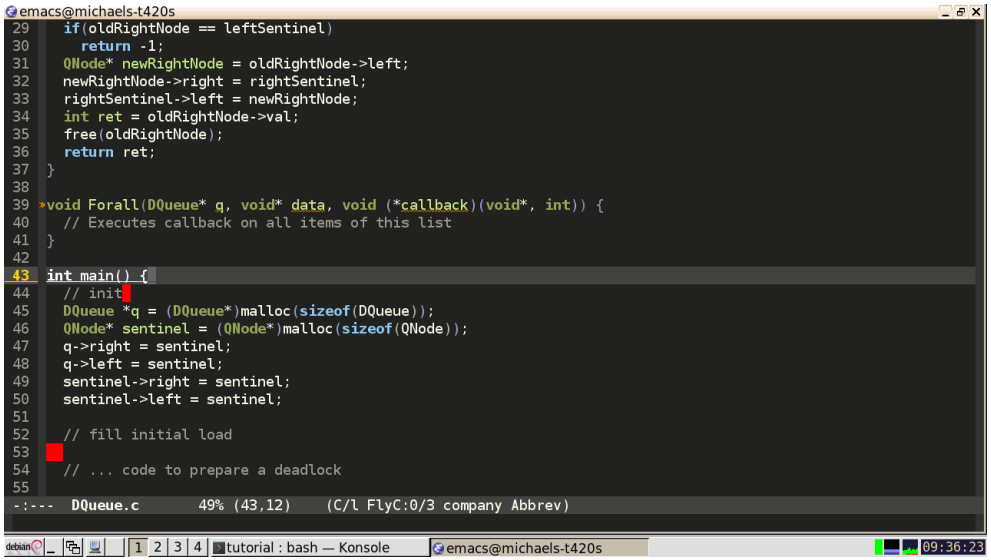

tutorial : bumper-semaphor — Konsole
Datei Bearbeiten Ansicht Lesezeichen Einstellungen Hilfe
bumper bumper.c bumper-fetchandadd.c bumper-semaphore.c dekker dekker.c~
bumper# bumper.c~ bumper-semaphore bumper-semaphore.c~ dekker.c DQueue.c
petter@michaels-t420s:/home/petter/lehre/tutorial$ time ./bumper
In main: creating thread 0
In main: creating thread 1
200000000
real 0m1.740s
user 0m3.380s
sys 0m0.000s
petter@michaels-t420s:/home/petter/lehre/tutorial$ timpe ./bumper-semaphore
Der Befehl »timpe« wurde nicht gefunden, meinten Sie vielleicht:
Befehl »time« aus dem Paket »time« (main)
timpe: Befehl nicht gefunden.
petter@michaels-t420s:/home/petter/lehre/tutorial$ time ./bumper-semaphore
In main: creating thread 0
In main: creating thread 1
200000000
real 0m33.418s
user 0m26.724s
sys 0m33.428s
petter@michaels-t420s:/home/petter/lehre/tutorial$
tutorial : bumper-semaphor
pthread_t threads[NUM_THREADS];
int rc;
long t;
sem_init(&sem,0,1);
-:--- bumper-semaphore.c 26% (29,20) (C/l FlyC:0/8 company Abbrev)
Wrote /home/petter/lehre/tutorial/bumper-semaphore.c
tutorial : bumper-se...
2locks : bash — Kon...
Programming Langu...
emacs@michaels-t4...
09:30:18

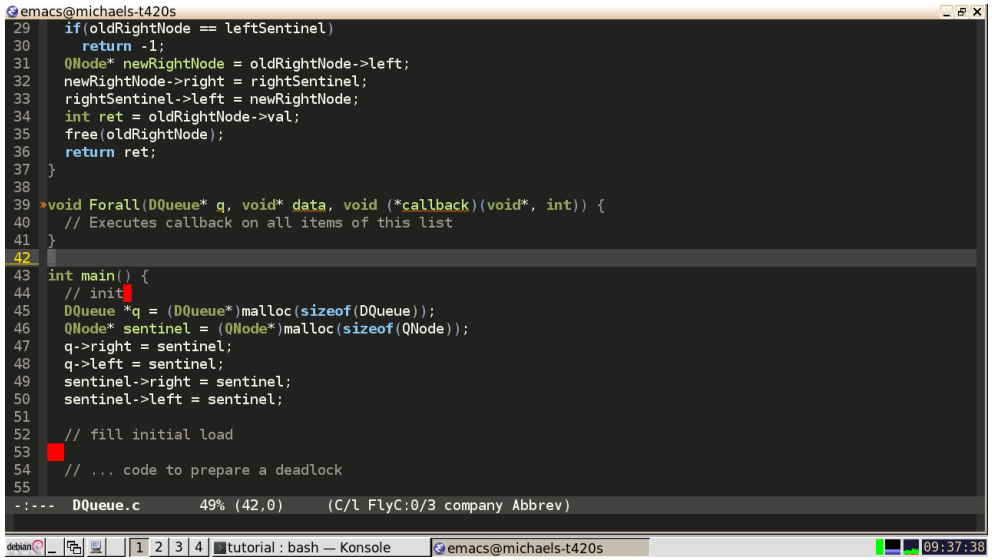

emacs@michaels-t420s
15 sem_wait(&sem);
16 //start critical section
17 start = firstFree;
18 firstFree = firstFree + size;
19 //end critical section
20 sem_post(&sem);
21
22 //if (start+size>sizeof(heap)) garbage_collect();
23 if (start+size>2^22) ;
24
25 return start;
26 }
27
28 void *allocator(void *threadid){
29 for (int i=0;i<100000000;i++)
30 myalloc(1);
31
32 }
33
34
35 int main(int argc, char *argv[]) {
36 pthread_t threads[NUM_THREADS];
37 int rc;
38 long t;
39
40 sem_init(&sem,0,1);
41
-:--- bumper-semaphore.c 26% (38,9) (C/l FlyC:0/8 company Abbrev)
tutorial : bash — Ko...
2locks : bash — Kon...
Programming Langu...
emacs@michaels-t4...
09:31:27

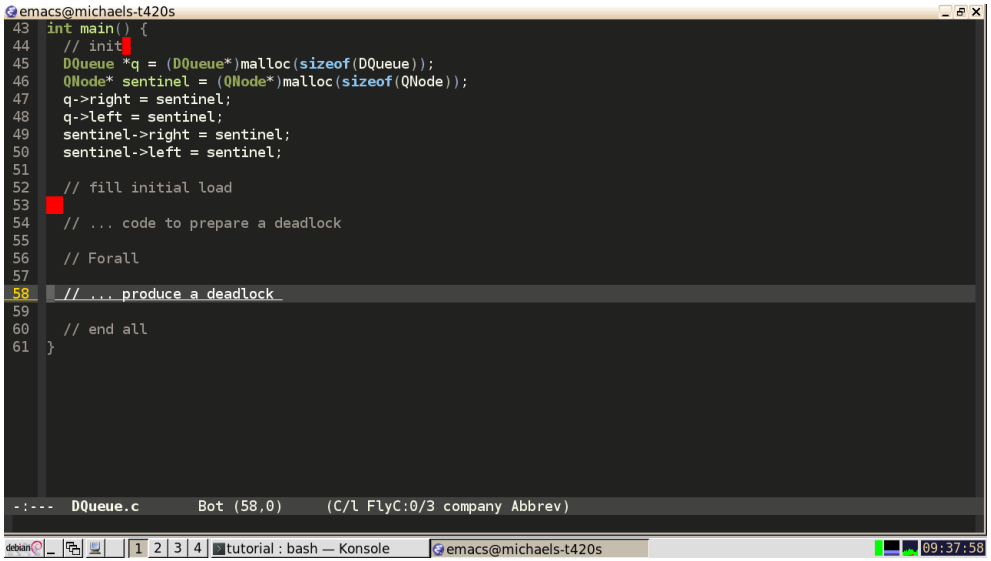

tutorial : bash — Konsole
Datei Bearbeiten Ansicht Lesezeichen Einstellungen Hilfe
bumper bumper.c bumper-fetchandadd.c bumper-semaphore.c dekker dekker.c~
bumper# bumper.c~ bumper-semaphore bumper-semaphore.c~ dekker.c DQueue.c
petter@michaels-t420s:/home/petter/lehre/tutorial$ time ./bumper
In main: creating thread 0
In main: creating thread 1
200000000
real 0m1.740s
user 0m3.380s
sys 0m0.000s
petter@michaels-t420s:/home/petter/lehre/tutorial$ timpe ./bumper-semaphore
Der Befehl »timpe« wurde nicht gefunden, meinten Sie vielleicht:
Befehl »time« aus dem Paket »time« (main)
timpe: Befehl nicht gefunden.
petter@michaels-t420s:/home/petter/lehre/tutorial$ time ./bumper-semaphore
In main: creating thread 0
In main: creating thread 1
200000000
real 0m33.418s
user 0m26.724s
sys 0m33.428s
petter@michaels-t420s:/home/petter/lehre/tutorial$
tutorial : bash
ensions/privateTab@infocatcher.xpi!/bootstrap.js:4188:11
before@resource://gre/modules/addons/XPIProvider.jsm -> jar:file:///home/petter/.mozilla/firefox/soz554wo.default/extensions/privateTab@infocatcher.xpi!/bootstrap.js:4154:23
wrapper@chrome://privatetab/content/patcher.jsm:36:16
getIndicatorData@resource://app/modules/DownloadsCommon.jsm:251:9
ensureTerminated@chrome://browser/content/downloads/indicator.js:234:5
onWindowUnload@chrome://browser/content/downloads/indicator.js:497:5
petter@michaels-t420s:/home/petter/lehre/proglang/slides/2locks$
2locks : bash
tutorial : bash — Konsole
2locks : bash — Konsole
Programming Languages – O...
09:31:43

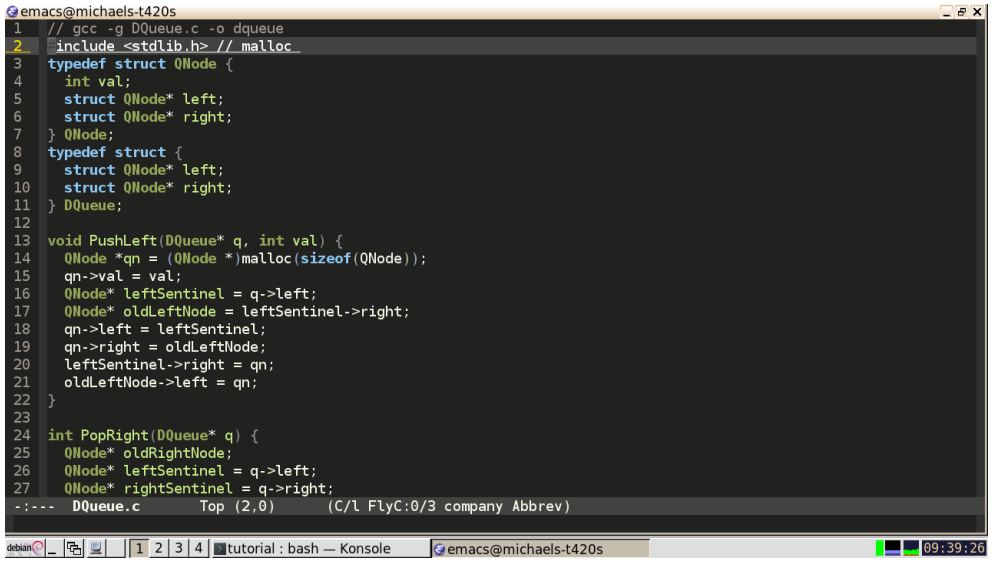

tutorial : bash — Konsole
Datei Bearbeiten Ansicht Lesezeichen Einstellungen Hilfe
real 0m1.740s
user 0m3.380s
sys 0m0.000s
petter@michaels-t420s:/home/petter/lehre/tutorial$ timpe
Der Befehl »timpe« wurde nicht gefunden, meinten Sie viel
Befehl »time« aus dem Paket »time« (main)
timpe: Befehl nicht gefunden.
petter@michaels-t420s:/home/petter/lehre/tutorial$ time .
In main: creating thread 0
In main: creating thread 1
200000000
real 0m33.418s
user 0m26.724s
sys 0m33.428s
petter@michaels-t420s:/home/petter/lehre/tutorial$ ls
bumper bumper.c bumper-fetchandadd.c bumper-semaphor
bumper# bumper.c~ bumper-semaphore bumper-semaphor
[1]+ Fertig emacs bumper.c
petter@michaels-t420s:/home/petter/lehre/tutorial$ emacs
[1] 3233
petter@michaels-t420s:/home/petter/lehre/tutorial$
tutorial : bash
emacs@michaels-t420s
tutorial : bash — Konsole
emacs@michaels-t420s
09:32:17

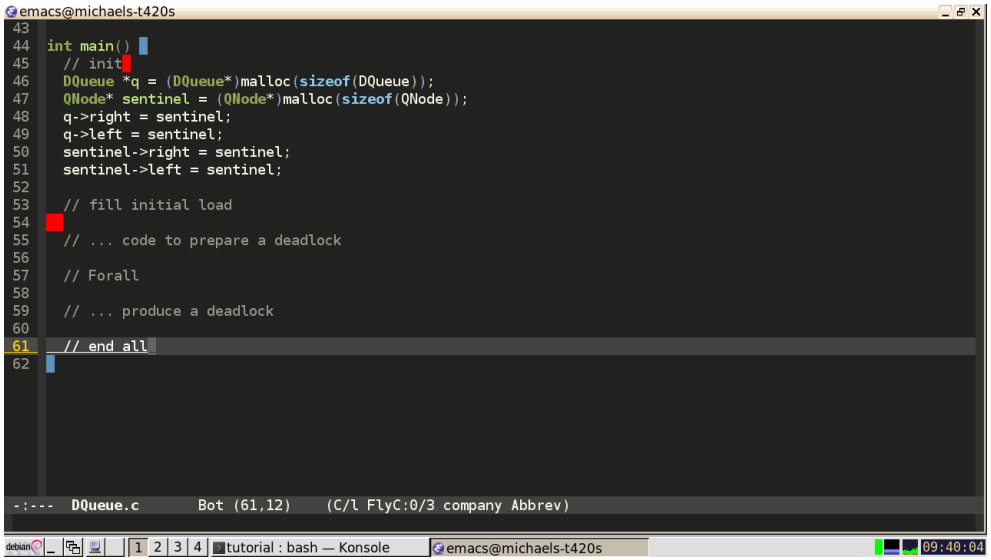

emacs@michaels-t420s
15 qn->val = val;
16 QNode* leftSentinel = q->left;
17 QNode* oldLeftNode = leftSentinel->right;
18 qn->left = leftSentinel;
19 qn->right = oldLeftNode;
20 leftSentinel->right = qn;
21 oldLeftNode->left = qn;
22 }
23
24 int PopRight(DQueue* q) {
25 QNode* oldRightNode;
26 QNode* leftSentinel = q->left;
27 QNode* rightSentinel = q->right;
28 oldRightNode = rightSentinel->left;
29 if(oldRightNode == leftSentinel)
30 return -1;
31 QNode* newRightNode = ol RightNode->left;
32 newRightNode->right = rightSentinel;
33 rightSentinel->left = newRightNode;
34 int ret = oldRightNode->val;
35 free(oldRightNode);
36 return ret;
37 }
38
39 void Forall(DQueue* q, void* data, void (*callback)(void*, int)) {
40 // Executes callback on all items of this list
41 }
-:--- DQueue.c 23% (31,26) (C/l FlyC:0/3 company Abbrev)
tutorial : bash — Konsole
emacs@michaels-t420s
09:35:19

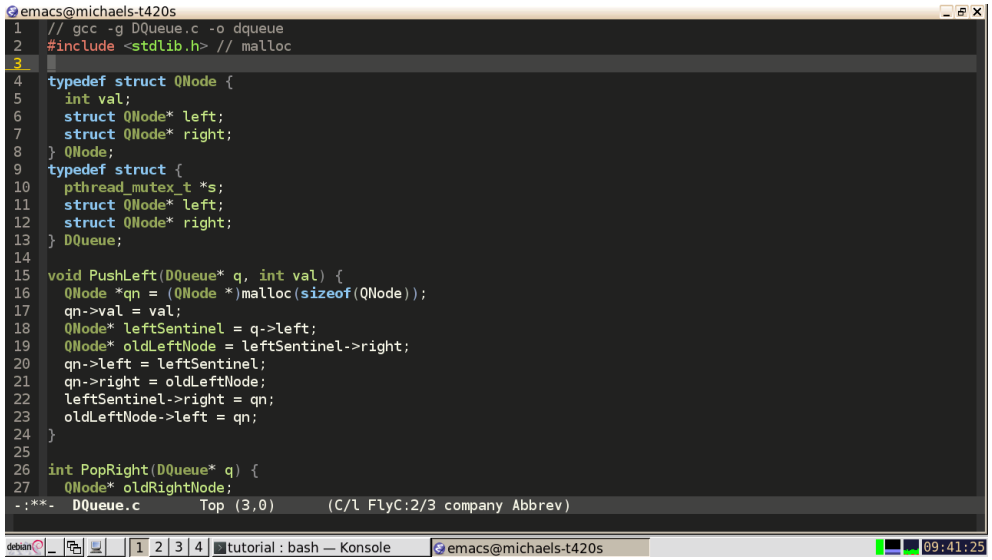

emacs@michaels-t420s
29 if(oldRightNode == leftSentinel)
30 return -1;
31 QNode* newRightNode = oldRightNode->left;
32 newRightNode->right = rightSentinel;
33 rightSentinel->left = newRightNode;
34 int ret = oldRightNode->val;
35 free(oldRightNode);
36 return ret;
37 }
38
39 void Forall(DQueue* q, void* data, void (*callback)(void*, int)) {
40 // Executes callback on all items of this list
41 }
42
43 int main() {
44 // init
45 DQueue *q = (DQueue*)malloc(sizeof(DQueue));
46 QNode* sentinel = (QNode*)malloc(sizeof(QNode));
47 q->right = sentinel;
48 q->left = sentinel;
49 sentinel->right = sentinel;
50 sentinel->left = sentinel;
51
52 // fill initial load
53
54 // ... code to prepare a deadlock
55
-:--- DQueue.c 49% (43,12) (C/l FlyC:0/3 company Abbrev)
1 2 3 4 tutorial : bash — Konsole emacs@michaels-t420s 09:36:23

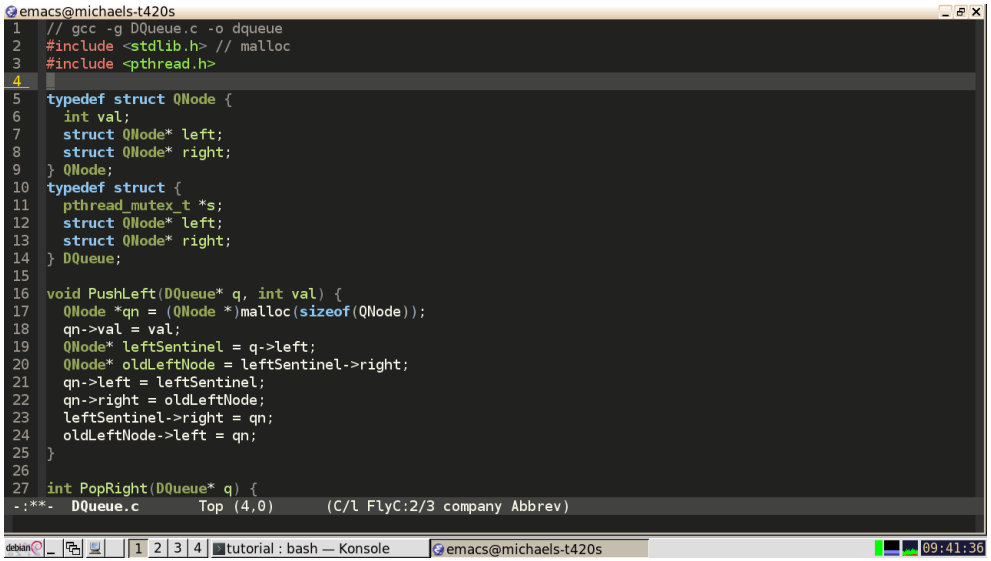

emacs@michaels-t420s
29 if(oldRightNode == leftSentinel)
30 return -1;
31 QNode* newRightNode = oldRightNode->left;
32 newRightNode->right = rightSentinel;
33 rightSentinel->left = newRightNode;
34 int ret = oldRightNode->val;
35 free(oldRightNode);
36 return ret;
37 }
38
39 void Forall(DQueue* q, void* data, void (*callback)(void*, int)) {
40 // Executes callback on all items of this list
41 }
42
43 int main() {
44 // init
45 DQueue *q = (DQueue*)malloc(sizeof(DQueue));
46 QNode* sentinel = (QNode*)malloc(sizeof(QNode));
47 q->right = sentinel;
48 q->left = sentinel;
49 sentinel->right = sentinel;
50 sentinel->left = sentinel;
51
52 // fill initial load
53
54 // ... code to prepare a deadlock
55
-:--- DQueue.c 49% (42,0) (C/l FlyC:0/3 company Abbrev)
1 2 3 4 tutorial : bash — Konsole emacs@michaels-t420s 09:37:38

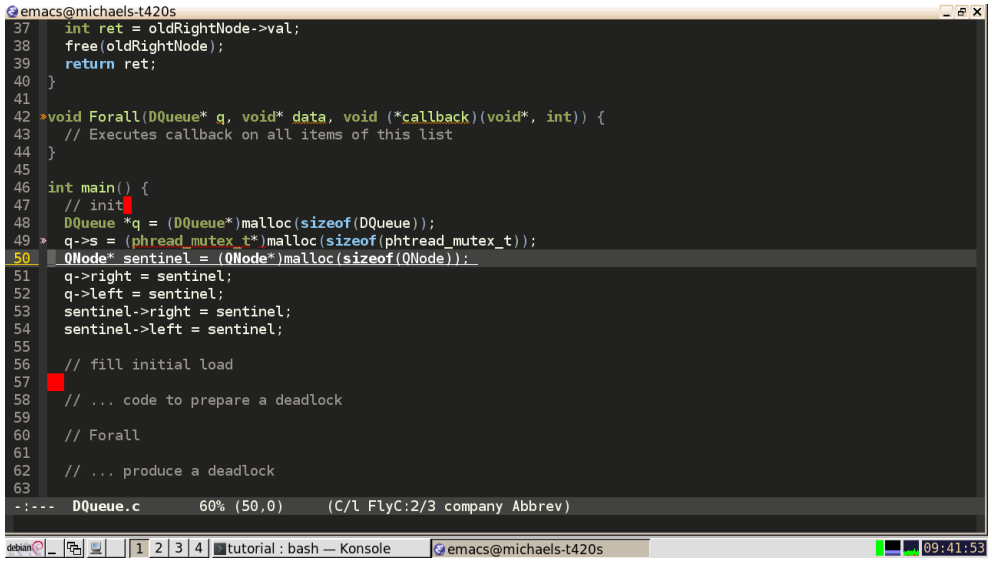

emacs@michaels-t420s
43 int main() {
44 // init
45 DQueue *q = (DQueue*)malloc(sizeof(DQueue));
46 QNode* sentinel = (QNode*)malloc(sizeof(QNode));
47 q->right = sentinel;
48 q->left = sentinel;
49 sentinel->right = sentinel;
50 sentinel->left = sentinel;
51
52 // fill initial load
53
54 // ... code to prepare a deadlock
55
56 // Forall
57
58 // ... produce a deadlock
59
60 // end all
61 }
-:--- DQueue.c Bot (58,0) (C/l FlyC:0/3 company Abbrev)
1 2 3 4 tutorial : bash — Konsole emacs@michaels-t420s 09:37:58

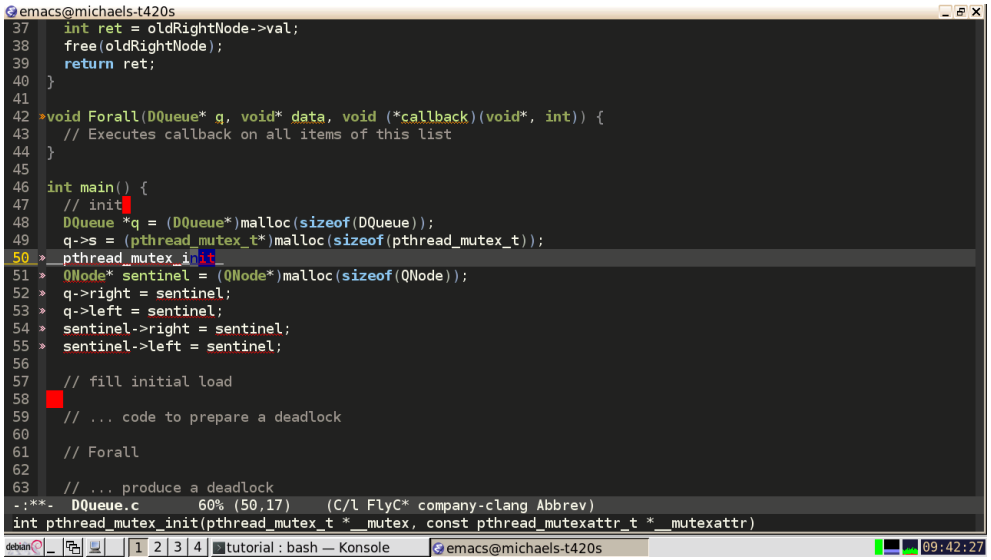

emacs@michaels-t420s
1 // gcc -g DQueue.c -o dqueue
2 #include <stdlib.h> // malloc
3 typedef struct QNode {
4 int val;
5 struct QNode* left;
6 struct QNode* right;
7 } QNode;
8 typedef struct {
9 struct QNode* left;
10 struct QNode* right;
11 } DQueue;
12
13 void PushLeft(DQueue* q, int val) {
14 QNode *qn = (QNode *)malloc(sizeof(QNode));
15 qn->val = val;
16 QNode* leftSentinel = q->left;
17 QNode* oldLeftNode = leftSentinel->right;
18 qn->left = leftSentinel;
19 qn->right = oldLeftNode;
20 leftSentinel->right = qn;
21 oldLeftNode->left = qn;
22 }
23
24 int PopRight(DQueue* q) {
25 QNode* oldRightNode;
26 QNode* leftSentinel = q->left;
27 QNode* rightSentinel = q->right;
-:--- DQueue.c Top (2,0) (C/l FlyC:0/3 company Abbrev)
1 2 3 4 tutorial : bash — Konsole emacs@michaels-t420s 09:39:26

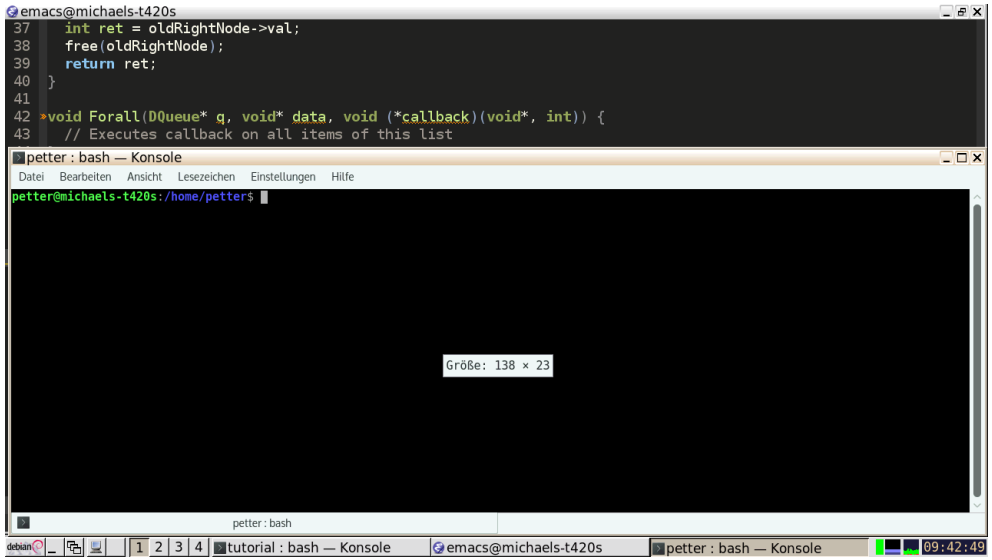

emacs@michaels-t420s
int main() {
// init
DQueue *q = (DQueue*)malloc(sizeof(DQueue));
QNode* sentinel = (QNode*)malloc(sizeof(QNode));
q->right = sentinel;
q->left = sentinel;
sentinel->right = sentinel;
sentinel->left = sentinel;
// fill initial load
// ... code to prepare a deadlock
// Forall
// ... produce a deadlock
// end all
-:--- DQueue.c Bot (61,12) (C/l FlyC:0/3 company Abbrev)
tutorial : bash — Konsole
emacs@michaels-t420s
09:40:04

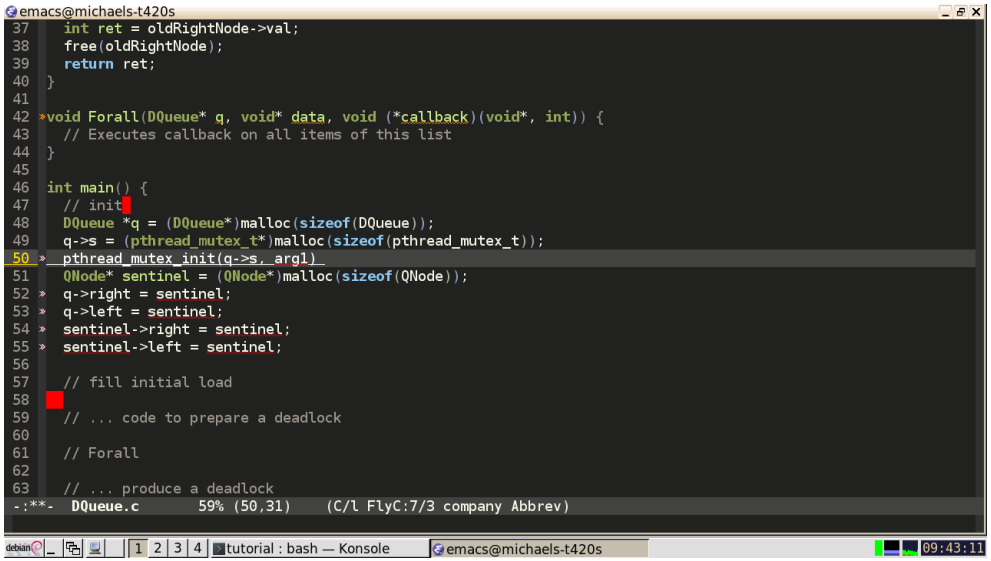

emacs@michaels-t420s
// gcc -g DQueue.c -o dqueue
#include <stdlib.h> // malloc
typedef struct QNode {
int val;
struct QNode* left;
struct QNode* right;
} QNode;
typedef struct {
pthread_mutex_t *s;
struct QNode* left;
struct QNode* right;
} DQueue;
void PushLeft(DQueue* q, int val) {
QNode *qn = (QNode *)malloc(sizeof(QNode));
qn->val = val;
QNode* leftSentinel = q->left;
QNode* oldLeftNode = leftSentinel->right;
qn->left = leftSentinel;
qn->right = oldLeftNode;
leftSentinel->right = qn;
oldLeftNode->left = qn;
}
int PopRight(DQueue* q) {
QNode* oldRightNode;
-:**- DQueue.c Top (3,0) (C/l FlyC:2/3 company Abbrev)
tutorial : bash — Konsole
emacs@michaels-t420s
09:41:25

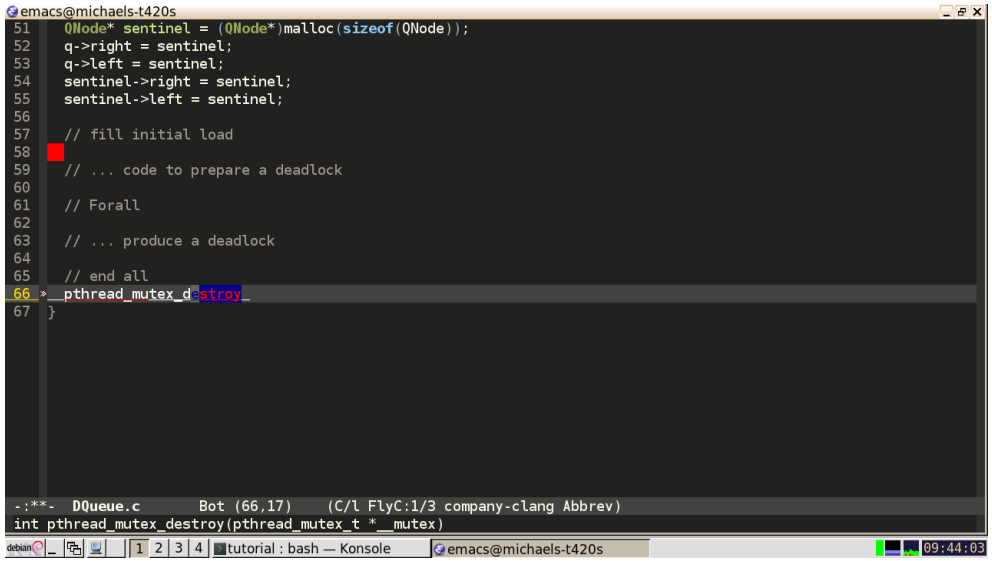

emacs@michaels-t420s
// gcc -g DQueue.c -o dqueue
#include <stdlib.h> // malloc
#include <pthread.h>
typedef struct QNode {
int val;
struct QNode* left;
struct QNode* right;
} QNode;
typedef struct {
pthread_mutex_t *s;
struct QNode* left;
struct QNode* right;
} DQueue;
void PushLeft(DQueue* q, int val) {
QNode *qn = (QNode *)malloc(sizeof(QNode));
qn->val = val;
QNode* leftSentinel = q->left;
QNode* oldLeftNode = leftSentinel->right;
qn->left = leftSentinel;
qn->right = oldLeftNode;
leftSentinel->right = qn;
oldLeftNode->left = qn;
}
int PopRight(DQueue* q) {
-:**- DQueue.c Top (4,0) (C/l FlyC:2/3 company Abbrev)
tutorial : bash — Konsole
emacs@michaels-t420s
09:41:36

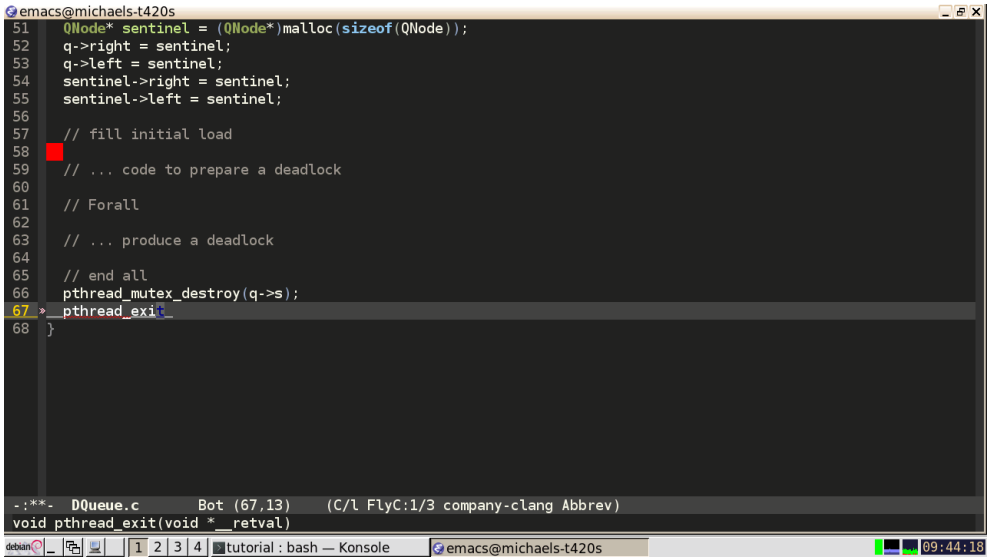

emacs@michaels-t420s
int ret = oldRightNode->val;
free(oldRightNode);
return ret;
}
void Forall(DQueue* q, void* data, void (*callback)(void*, int)) {
// Executes callback on all items of this list
}
int main() {
// init
DQueue *q = (DQueue*)malloc(sizeof(DQueue));
q->s = (phread_mutex_t*)malloc(sizeof(phtread_mutex_t));
QNode* sentinel = (QNode*)malloc(sizeof(QNode));
q->right = sentinel;
q->left = sentinel;
sentinel->right = sentinel;
sentinel->left = sentinel;
// fill initial load
// ... code to prepare a deadlock
// Forall
// ... produce a deadlock
-:--- DQueue.c 60% (50,0) (C/l FlyC:2/3 company Abbrev)
tutorial : bash — Konsole
emacs@michaels-t420s
09:41:53

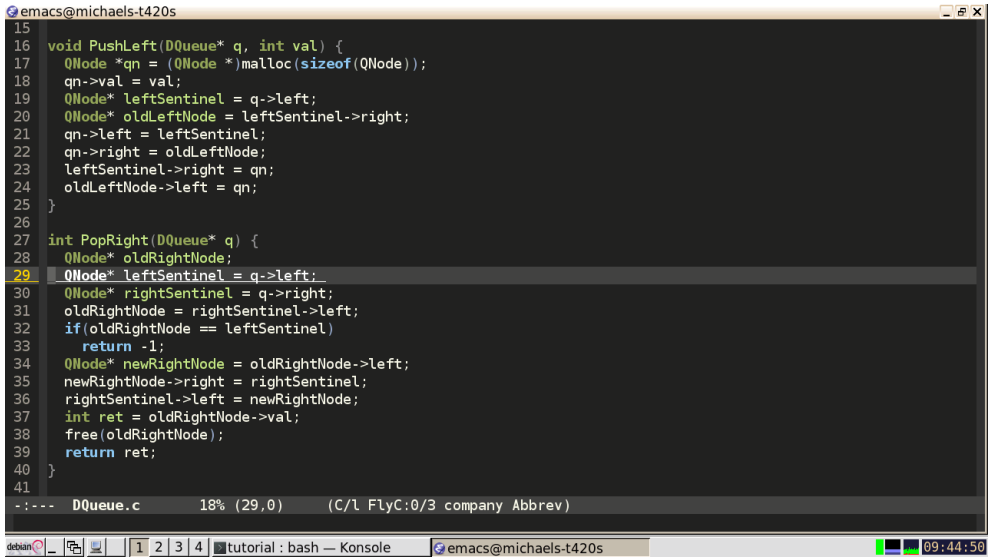

emacs@michaels-t420s
37 int ret = oldRightNode->val;
38 free(oldRightNode);
39 return ret;
40 }
41
42 »void Forall(DQueue* q, void* data, void (*callback)(void*, int)) {
43 // Executes callback on all items of this list
44 }
45
46 int main() {
47 // init
48 DQueue *q = (DQueue*)malloc(sizeof(DQueue));
49 q->s = (pthread_mutex_t*)malloc(sizeof(pthread_mutex_t));
50 »__pthread_mutex_init_
51 » QNode* sentinel = (QNode*)malloc(sizeof(QNode));
52 » q->right = sentinel;
53 » q->left = sentinel;
54 » sentinel->right = sentinel;
55 » sentinel->left = sentinel;
56
57 // fill initial load
58
59 // ... code to prepare a deadlock
60
61 // Forall
62
63 // ... produce a deadlock
-:**- DQueue.c 60% (50,17) (C/l FlyC* company-clang Abbrev)
int pthread_mutex_init(pthread_mutex_t *__mutex, const pthread_mutexattr_t *__mutexattr)
1 2 3 4 tutorial : bash — Konsole emacs@michaels-t420s 09:42:27

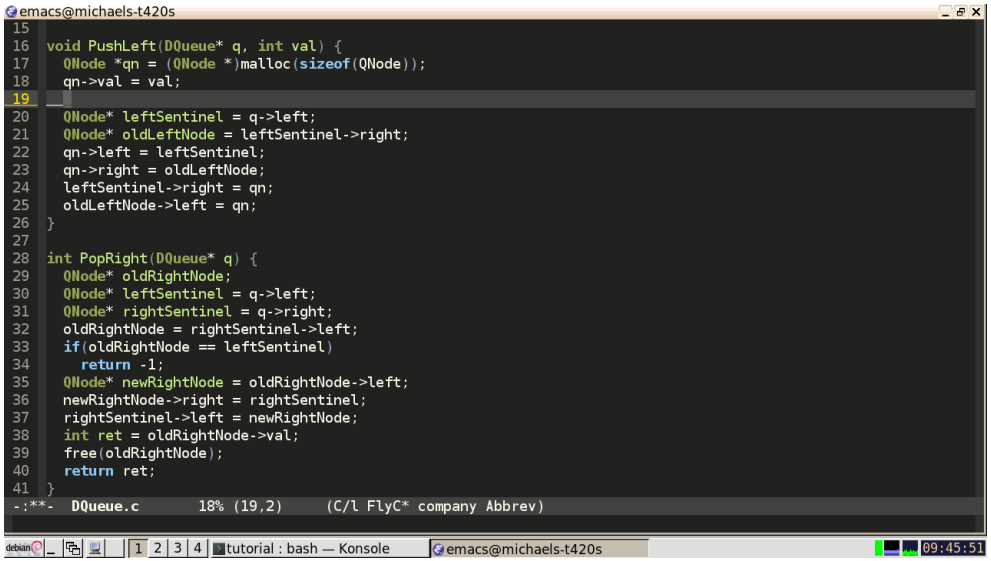

emacs@michaels-t420s
37 int ret = oldRightNode->val;
38 free(oldRightNode);
39 return ret;
40 }
41
42 »void Forall(DQueue* q, void* data, void (*callback)(void*, int)) {
43 // Executes callback on all items of this list
petter : bash — Konsole
Datei Bearbeiten Ansicht Lesezeichen Einstellungen Hilfe
petter@michaels-t420s:/home/petter$
Größe: 138 × 23
petter : bash
1 2 3 4 tutorial : bash — Konsole emacs@michaels-t420s petter : bash — Konsole 09:42:49

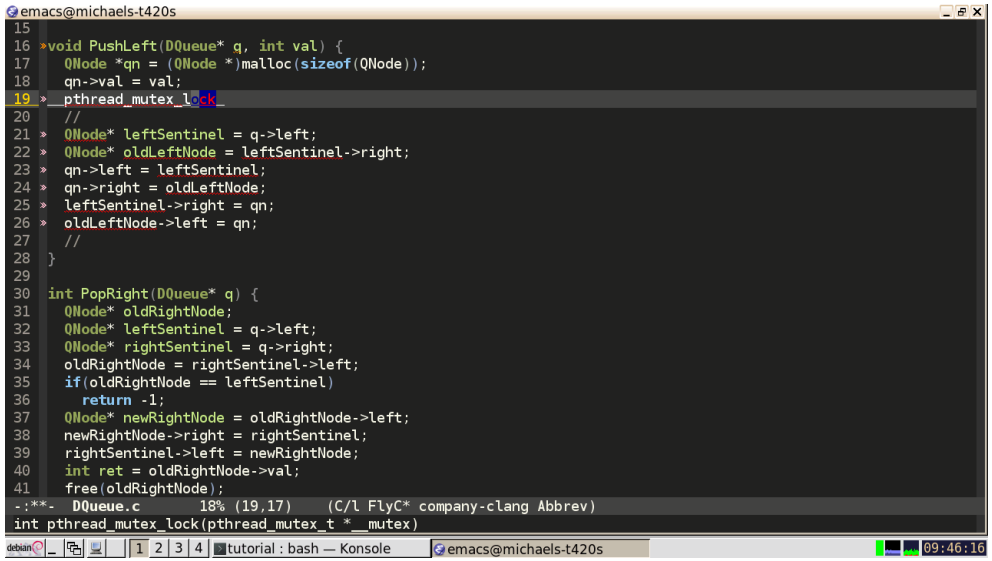

emacs@michaels-t420s
37 int ret = oldRightNode->val;
38 free(oldRightNode);
39 return ret;
40 }
41
42 »void Forall(DQueue* q, void* data, void (*callback)(void*, int)) {
43 // Executes callback on all items of this list
44 }
45
46 int main() {
47 // init
48 DQueue *q = (DQueue*)malloc(sizeof(DQueue));
49 q->s = (pthread_mutex_t*)malloc(sizeof(pthread_mutex_t));
50 »__pthread_mutex_init(q->s, _arg1)
51 QNode* sentinel = (QNode*)malloc(sizeof(QNode));
52 » q->right = sentinel;
53 » q->left = sentinel;
54 » sentinel->right = sentinel;
55 » sentinel->left = sentinel;
56
57 // fill initial load
58
59 // ... code to prepare a deadlock
60
61 // Forall
62
63 // ... produce a deadlock
-:**- DQueue.c 59% (50,31) (C/l FlyC:7/3 company Abbrev)
1 2 3 4 tutorial : bash — Konsole emacs@michaels-t420s 09:43:11

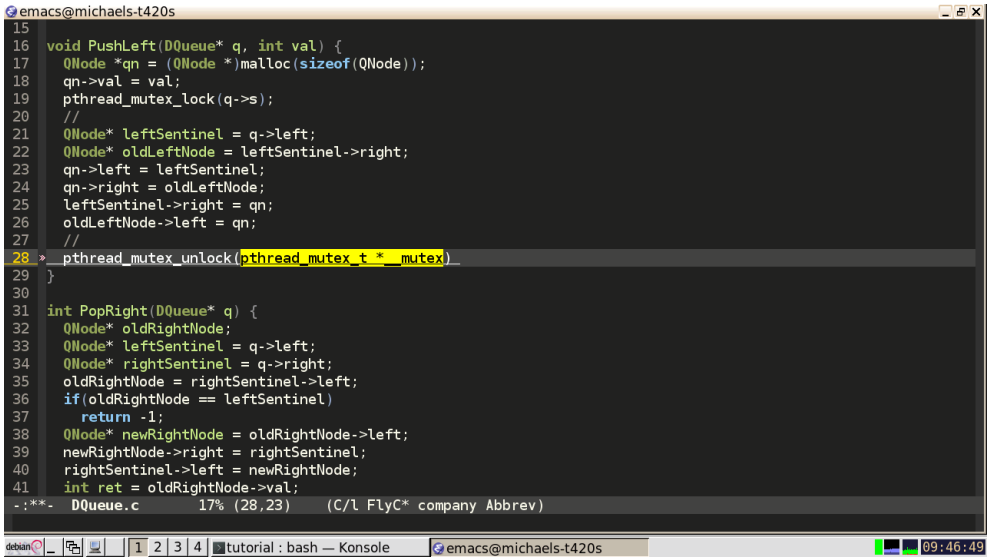

emacs@michaels-t420s
51 QNode* sentinel = (QNode*)malloc(sizeof(QNode));
52 q->right = sentinel;
53 q->left = sentinel;
54 sentinel->right = sentinel;
55 sentinel->left = sentinel;
56
57 // fill initial load
58
59 // ... code to prepare a deadlock
60
61 // Forall
62
63 // ... produce a deadlock
64
65 // end all
66 »__pthread_mutex_destroy_
67 }
-:**- DQueue.c Bot (66,17) (C/l FlyC:1/3 company-clang Abbrev)
int pthread_mutex_destroy(pthread_mutex_t *__mutex)
1 2 3 4 tutorial : bash — Konsole emacs@michaels-t420s 09:44:03

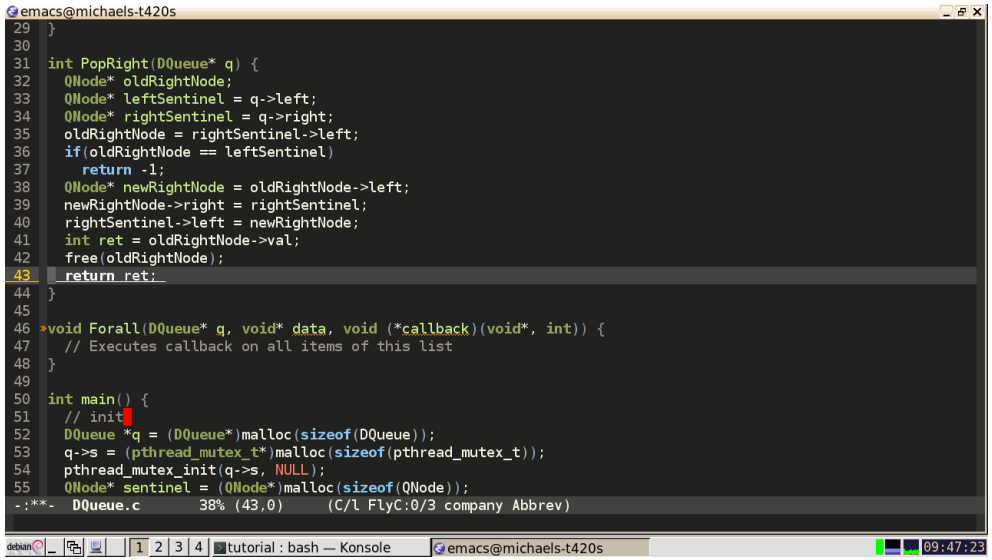

emacs@michaels-t420s
QNode* sentinel = (QNode*)malloc(sizeof(QNode));
q->right = sentinel;
q->left = sentinel;
sentinel->right = sentinel;
sentinel->left = sentinel;
// fill initial load
// ... code to prepare a deadlock
// Forall
// ... produce a deadlock
// end all
pthread_mutex_destroy(q->s);
__pthread_exit
}
-:**- DQueue.c Bot (67,13) (C/l FlyC:1/3 company-clang Abbrev)
void pthread_exit(void *__retval)
tutorial : bash — Konsole
emacs@michaels-t420s
09:44:18

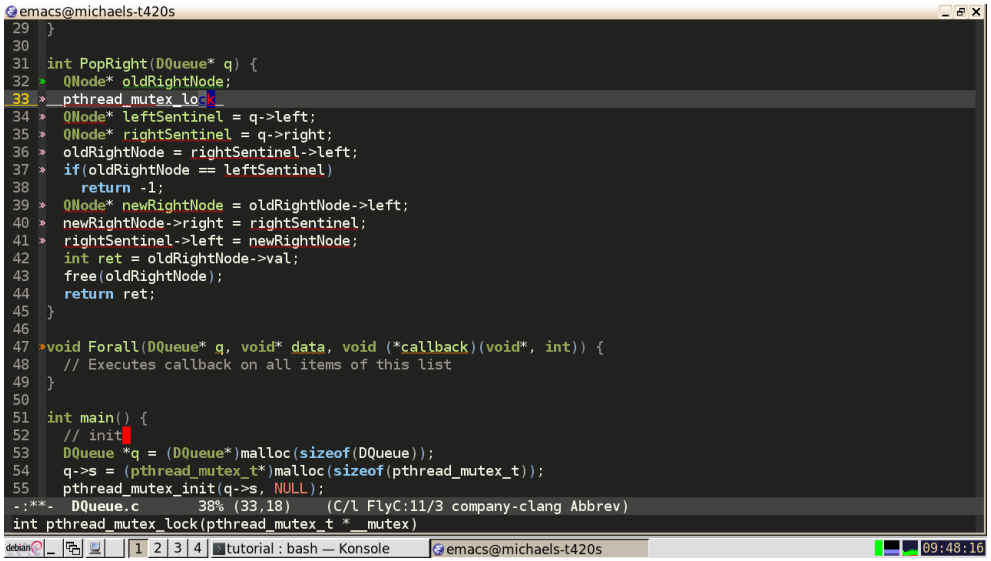

emacs@michaels-t420s
void PushLeft(DQueue* q, int val) {
QNode *qn = (QNode *)malloc(sizeof(QNode));
qn->val = val;
QNode* leftSentinel = q->left;
QNode* oldLeftNode = leftSentinel->right;
qn->left = leftSentinel;
qn->right = oldLeftNode;
leftSentinel->right = qn;
oldLeftNode->left = qn;
}
int PopRight(DQueue* q) {
QNode* oldRightNode;
QNode* leftSentinel = q->left;
QNode* rightSentinel = q->right;
oldRightNode = rightSentinel->left;
if(oldRightNode == leftSentinel)
return -1;
QNode* newRightNode = oldRightNode->left;
newRightNode->right = rightSentinel;
rightSentinel->left = newRightNode;
int ret = oldRightNode->val;
free(oldRightNode);
return ret;
}
-:--- DQueue.c 18% (29,0) (C/l FlyC:0/3 company Abbrev)
tutorial : bash — Konsole
emacs@michaels-t420s
09:44:50

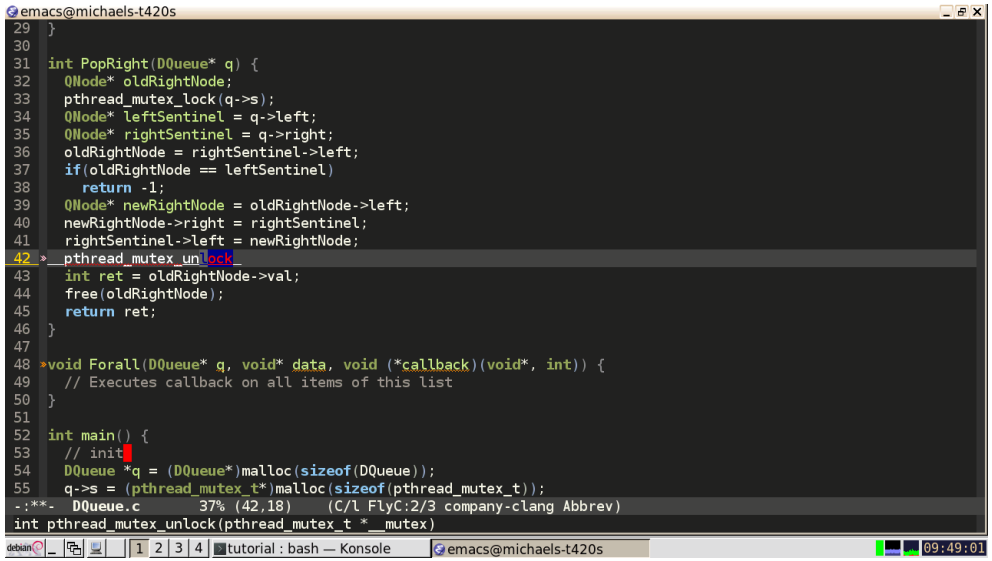

emacs@michaels-t420s
void PushLeft(DQueue* q, int val) {
QNode *qn = (QNode *)malloc(sizeof(QNode));
qn->val = val;
QNode* leftSentinel = q->left;
QNode* oldLeftNode = leftSentinel->right;
qn->left = leftSentinel;
qn->right = oldLeftNode;
leftSentinel->right = qn;
oldLeftNode->left = qn;
}
int PopRight(DQueue* q) {
QNode* oldRightNode;
QNode* leftSentinel = q->left;
QNode* rightSentinel = q->right;
oldRightNode = rightSentinel->left;
if(oldRightNode == leftSentinel)
return -1;
QNode* newRightNode = oldRightNode->left;
newRightNode->right = rightSentinel;
rightSentinel->left = newRightNode;
int ret = oldRightNode->val;
free(oldRightNode);
return ret;
}
-:**- DQueue.c 18% (19,2) (C/l FlyC* company Abbrev)
tutorial : bash — Konsole
emacs@michaels-t420s
09:45:51

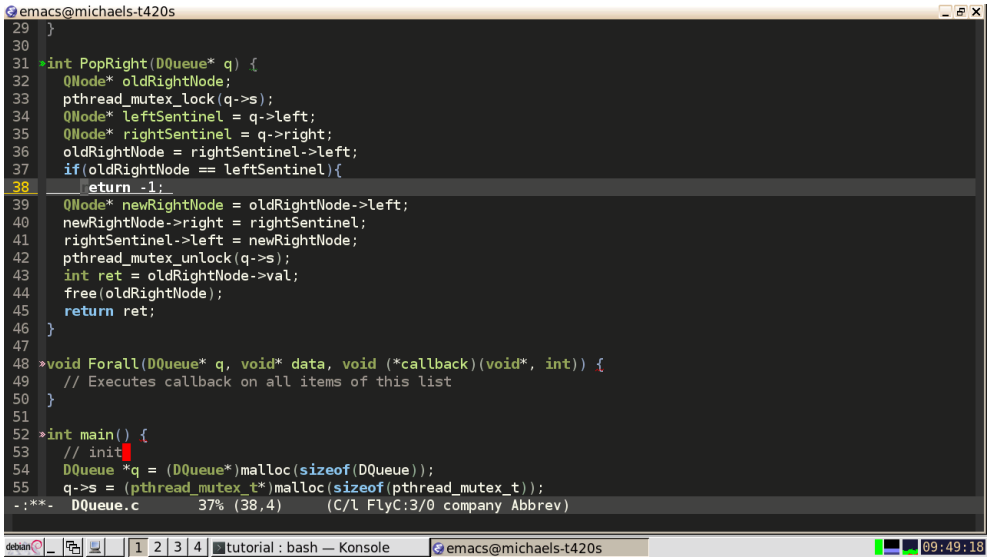

emacs@michaels-t420s
void PushLeft(DQueue* q, int val) {
QNode *qn = (QNode *)malloc(sizeof(QNode));
qn->val = val;
__pthread_mutex_lock
//
QNode* leftSentinel = q->left;
QNode* oldLeftNode = leftSentinel->right;
qn->left = leftSentinel;
qn->right = oldLeftNode;
leftSentinel->right = qn;
oldLeftNode->left = qn;
//
}
int PopRight(DQueue* q) {
QNode* oldRightNode;
QNode* leftSentinel = q->left;
QNode* rightSentinel = q->right;
oldRightNode = rightSentinel->left;
if(oldRightNode == leftSentinel)
return -1;
QNode* newRightNode = oldRightNode->left;
newRightNode->right = rightSentinel;
rightSentinel->left = newRightNode;
int ret = oldRightNode->val;
free(oldRightNode);
-:**- DQueue.c 18% (19,17) (C/l FlyC* company-clang Abbrev)
int pthread_mutex_lock(pthread_mutex_t *__mutex)
tutorial : bash — Konsole
emacs@michaels-t420s
09:46:16

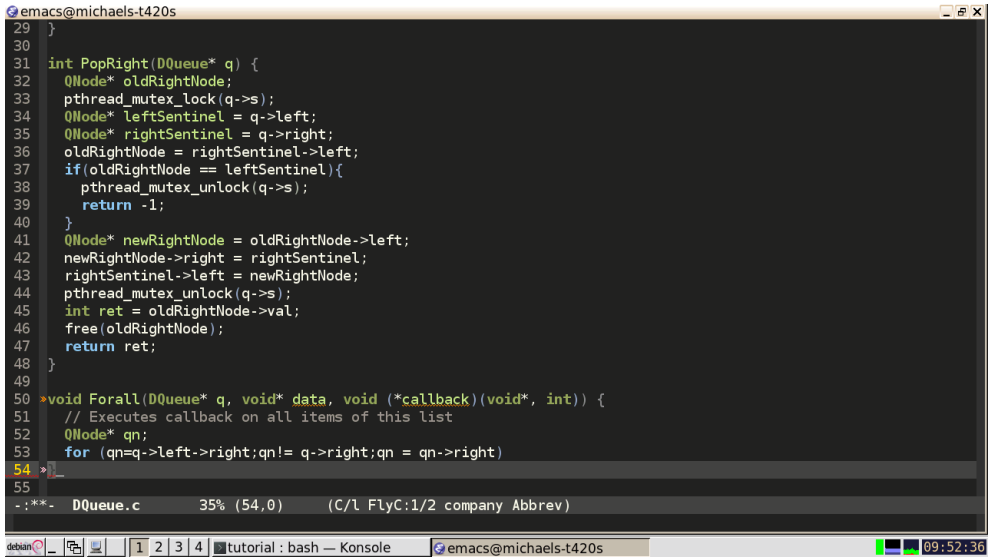

emacs@michaels-t420s
void PushLeft(DQueue* q, int val) {
QNode *qn = (QNode *)malloc(sizeof(QNode));
qn->val = val;
pthread_mutex_lock(q->s);
//
QNode* leftSentinel = q->left;
QNode* oldLeftNode = leftSentinel->right;
qn->left = leftSentinel;
qn->right = oldLeftNode;
leftSentinel->right = qn;
oldLeftNode->left = qn;
//
pthread_mutex_unlock(pthread_mutex_t *__mutex)
}
int PopRight(DQueue* q) {
QNode* oldRightNode;
QNode* leftSentinel = q->left;
QNode* rightSentinel = q->right;
oldRightNode = rightSentinel->left;
if(oldRightNode == leftSentinel)
return -1;
QNode* newRightNode = oldRightNode->left;
newRightNode->right = rightSentinel;
rightSentinel->left = newRightNode;
int ret = oldRightNode->val;
-:**- DQueue.c 17% (28,23) (C/l FlyC* company Abbrev)
tutorial : bash — Konsole
emacs@michaels-t420s
09:46:49

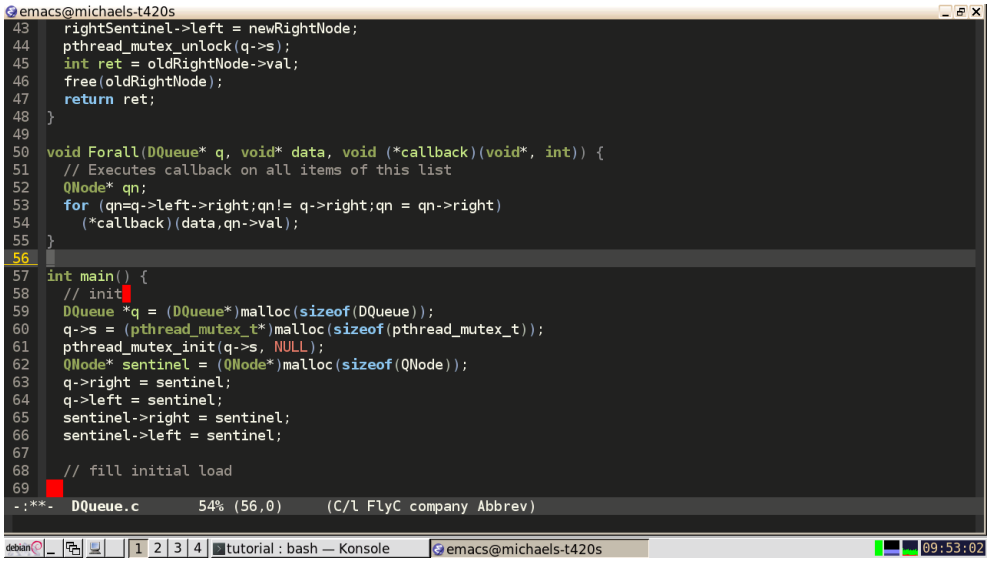

emacs@michaels-t420s
}
int PopRight(DQueue* q) {
QNode* oldRightNode;
QNode* leftSentinel = q->left;
QNode* rightSentinel = q->right;
oldRightNode = rightSentinel->left;
if(oldRightNode == leftSentinel)
return -1;
QNode* newRightNode = oldRightNode->left;
newRightNode->right = rightSentinel;
rightSentinel->left = newRightNode;
int ret = oldRightNode->val;
free(oldRightNode);
return ret;
}
void Forall(DQueue* q, void* data, void (*callback)(void*, int)) {
// Executes callback on all items of this list
}
int main() {
// init
DQueue *q = (DQueue*)malloc(sizeof(DQueue));
q->s = (pthread_mutex_t*)malloc(sizeof(pthread_mutex_t));
pthread_mutex_init(q->s, NULL);
QNode* sentinel = (QNode*)malloc(sizeof(QNode));
-:**- DQueue.c 38% (43,0) (C/l FlyC:0/3 company Abbrev)
tutorial : bash — Konsole
emacs@michaels-t420s
09:47:23

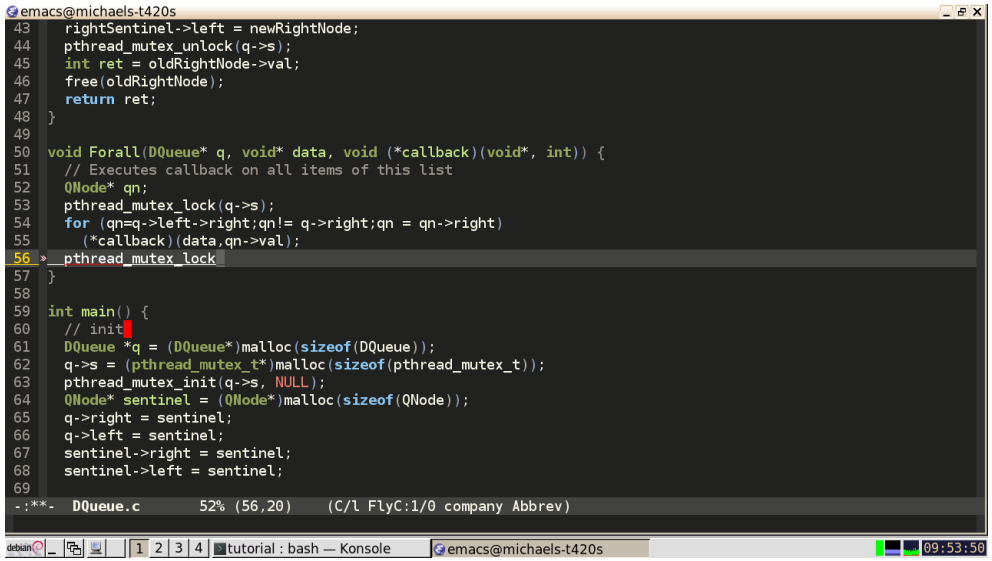

emacs@michaels-t420s
}
int PopRight(DQueue* q) {
QNode* oldRightNode;
pthread_mutex_lock
QNode* leftSentinel = q->left;
QNode* rightSentinel = q->right;
oldRightNode = rightSentinel->left;
if(oldRightNode == leftSentinel)
return -1;
QNode* newRightNode = oldRightNode->left;
newRightNode->right = rightSentinel;
rightSentinel->left = newRightNode;
int ret = oldRightNode->val;
free(oldRightNode);
return ret;
}
void Forall(DQueue* q, void* data, void (*callback)(void*, int)) {
// Executes callback on all items of this list
}
int main() {
// init
DQueue *q = (DQueue*)malloc(sizeof(DQueue));
q->s = (pthread_mutex_t*)malloc(sizeof(pthread_mutex_t));
pthread_mutex_init(q->s, NULL);
-:**- DQueue.c 38% (33,18) (C/l FlyC:11/3 company-clang Abbrev)
int pthread_mutex_lock(pthread_mutex_t *__mutex)
tutorial : bash — Konsole
emacs@michaels-t420s
09:48:16

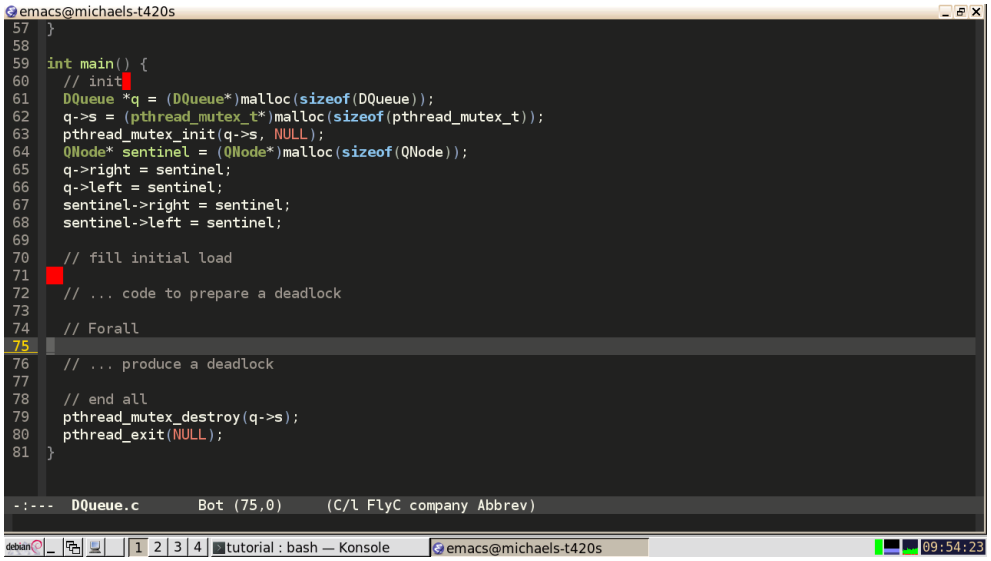

emacs@michaels-t420s
}
int PopRight(DQueue* q) {
QNode* oldRightNode;
pthread_mutex_lock(q->s);
QNode* leftSentinel = q->left;
QNode* rightSentinel = q->right;
oldRightNode = rightSentinel->left;
if(oldRightNode == leftSentinel)
return -1;
QNode* newRightNode = oldRightNode->left;
newRightNode->right = rightSentinel;
rightSentinel->left = newRightNode;
pthread_mutex_unlock
int ret = oldRightNode->val;
free(oldRightNode);
return ret;
}
void Forall(DQueue* q, void* data, void (*callback)(void*, int)) {
// Executes callback on all items of this list
}
int main() {
// init
DQueue *q = (DQueue*)malloc(sizeof(DQueue));
q->s = (pthread_mutex_t*)malloc(sizeof(pthread_mutex_t));
-:**- DQueue.c 37% (42,18) (C/l FlyC:2/3 company-clang Abbrev)
int pthread_mutex_unlock(pthread_mutex_t *__mutex)
tutorial : bash — Konsole
emacs@michaels-t420s
09:49:01

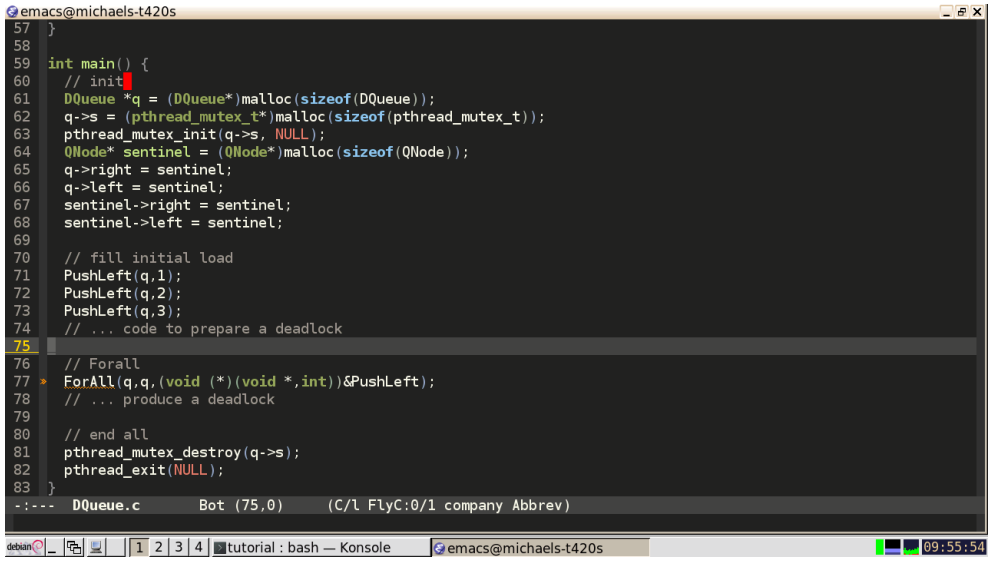

emacs@michaels-t420s
}
int PopRight(DQueue* q) {
QNode* oldRightNode;
pthread_mutex_lock(q->s);
QNode* leftSentinel = q->left;
QNode* rightSentinel = q->right;
oldRightNode = rightSentinel->left;
if(oldRightNode == leftSentinel){
return -1;
QNode* newRightNode = oldRightNode->left;
newRightNode->right = rightSentinel;
rightSentinel->left = newRightNode;
pthread_mutex_unlock(q->s);
int ret = oldRightNode->val;
free(oldRightNode);
return ret;
}
void Forall(DQueue* q, void* data, void (*callback)(void*, int)) {
// Executes callback on all items of this list
}
int main() {
// init
DQueue *q = (DQueue*)malloc(sizeof(DQueue));
q->s = (pthread_mutex_t*)malloc(sizeof(pthread_mutex_t));
-:**- DQueue.c 37% (38,4) (C/l FlyC:3/0 company Abbrev)
1 2 3 4
tutorial : bash — Konsole
emacs@michaels-t420s
09:49:18


emacs@michaels-t420s
}
int PopRight(DQueue* q) {
QNode* oldRightNode;
pthread_mutex_lock(q->s);
QNode* leftSentinel = q->left;
QNode* rightSentinel = q->right;
oldRightNode = rightSentinel->left;
if(oldRightNode == leftSentinel){
pthread_mutex_unlock(q->s);
return -1;
}
QNode* newRightNode = oldRightNode->left;
newRightNode->right = rightSentinel;
rightSentinel->left = newRightNode;
pthread_mutex_unlock(q->s);
int ret = oldRightNode->val;
free(oldRightNode);
return ret;
}
void Forall(DQueue* q, void* data, void (*callback)(void*, int)) {
// Executes callback on all items of this list
QNode* qn;
for (qn=q->left->right;qn!= q->right;qn = qn->right)
-:**- DQueue.c 35% (54,0) (C/l FlyC:1/2 company Abbrev)
1 2 3 4
tutorial : bash — Konsole
emacs@michaels-t420s
09:52:36


emacs@michaels-t420s
rightSentinel->left = newRightNode;
pthread_mutex_unlock(q->s);
int ret = oldRightNode->val;
free(oldRightNode);
return ret;
}
void Forall(DQueue* q, void* data, void (*callback)(void*, int)) {
// Executes callback on all items of this list
QNode* qn;
for (qn=q->left->right;qn!= q->right;qn = qn->right)
(*callback)(data,qn->val);
}
int main() {
// init
DQueue *q = (DQueue*)malloc(sizeof(DQueue));
q->s = (pthread_mutex_t*)malloc(sizeof(pthread_mutex_t));
pthread_mutex_init(q->s, NULL);
QNode* sentinel = (QNode*)malloc(sizeof(QNode));
q->right = sentinel;
q->left = sentinel;
sentinel->right = sentinel;
sentinel->left = sentinel;
// fill initial load
-:**- DQueue.c 54% (56,0) (C/l FlyC company Abbrev)
1 2 3 4
tutorial : bash — Konsole
emacs@michaels-t420s
09:53:02


emacs@michaels-t420s
rightSentinel->left = newRightNode;
pthread_mutex_unlock(q->s);
int ret = oldRightNode->val;
free(oldRightNode);
return ret;
}
void Forall(DQueue* q, void* data, void (*callback)(void*, int)) {
// Executes callback on all items of this list
QNode* qn;
pthread_mutex_lock(q->s);
for (qn=q->left->right;qn!= q->right;qn = qn->right)
(*callback)(data,qn->val);
pthread_mutex_lock
}
int main() {
// init
DQueue *q = (DQueue*)malloc(sizeof(DQueue));
q->s = (pthread_mutex_t*)malloc(sizeof(pthread_mutex_t));
pthread_mutex_init(q->s, NULL);
QNode* sentinel = (QNode*)malloc(sizeof(QNode));
q->right = sentinel;
q->left = sentinel;
sentinel->right = sentinel;
sentinel->left = sentinel;
-:**- DQueue.c 52% (56,20) (C/l FlyC:1/0 company Abbrev)
1 2 3 4
tutorial : bash — Konsole
emacs@michaels-t420s
09:53:50


emacs@michaels-t420s
57 }
58
59 int main() {
60 // init
61 DQueue *q = (DQueue*)malloc(sizeof(DQueue));
62 q->s = (pthread_mutex_t*)malloc(sizeof(pthread_mutex_t));
63 pthread_mutex_init(q->s, NULL);
64 QNode* sentinel = (QNode*)malloc(sizeof(QNode));
65 q->right = sentinel;
66 q->left = sentinel;
67 sentinel->right = sentinel;
68 sentinel->left = sentinel;
69
70 // fill initial load
71
72 // ... code to prepare a deadlock
73
74 // Forall
75
76 // ... produce a deadlock
77
78 // end all
79 pthread_mutex_destroy(q->s);
80 pthread_exit(NULL);
81 }
-:--- DQueue.c Bot (75,0) (C/l FlyC company Abbrev)
1 2 3 4 tutorial : bash — Konsole emacs@michaels-t420s 09:54:23


emacs@michaels-t420s
57 }
58
59 int main() {
60 // init
61 DQueue *q = (DQueue*)malloc(sizeof(DQueue));
62 q->s = (pthread_mutex_t*)malloc(sizeof(pthread_mutex_t));
63 pthread_mutex_init(q->s, NULL);
64 QNode* sentinel = (QNode*)malloc(sizeof(QNode));
65 q->right = sentinel;
66 q->left = sentinel;
67 sentinel->right = sentinel;
68 sentinel->left = sentinel;
69
70 // fill initial load
71 PushLeft(q,1);
72 PushLeft(q,2);
73 PushLeft(q,3);
74 // ... code to prepare a deadlock
75
76 // Forall
77 ForAll(q,q,(void (*)(void *,int))&PushLeft);
78 // ... produce a deadlock
79
80 // end all
81 pthread_mutex_destroy(q->s);
82 pthread_exit(NULL);
83 }
-:--- DQueue.c Bot (75,0) (C/l FlyC:0/1 company Abbrev)
1 2 3 4 tutorial : bash — Konsole emacs@michaels-t420s 09:55:54