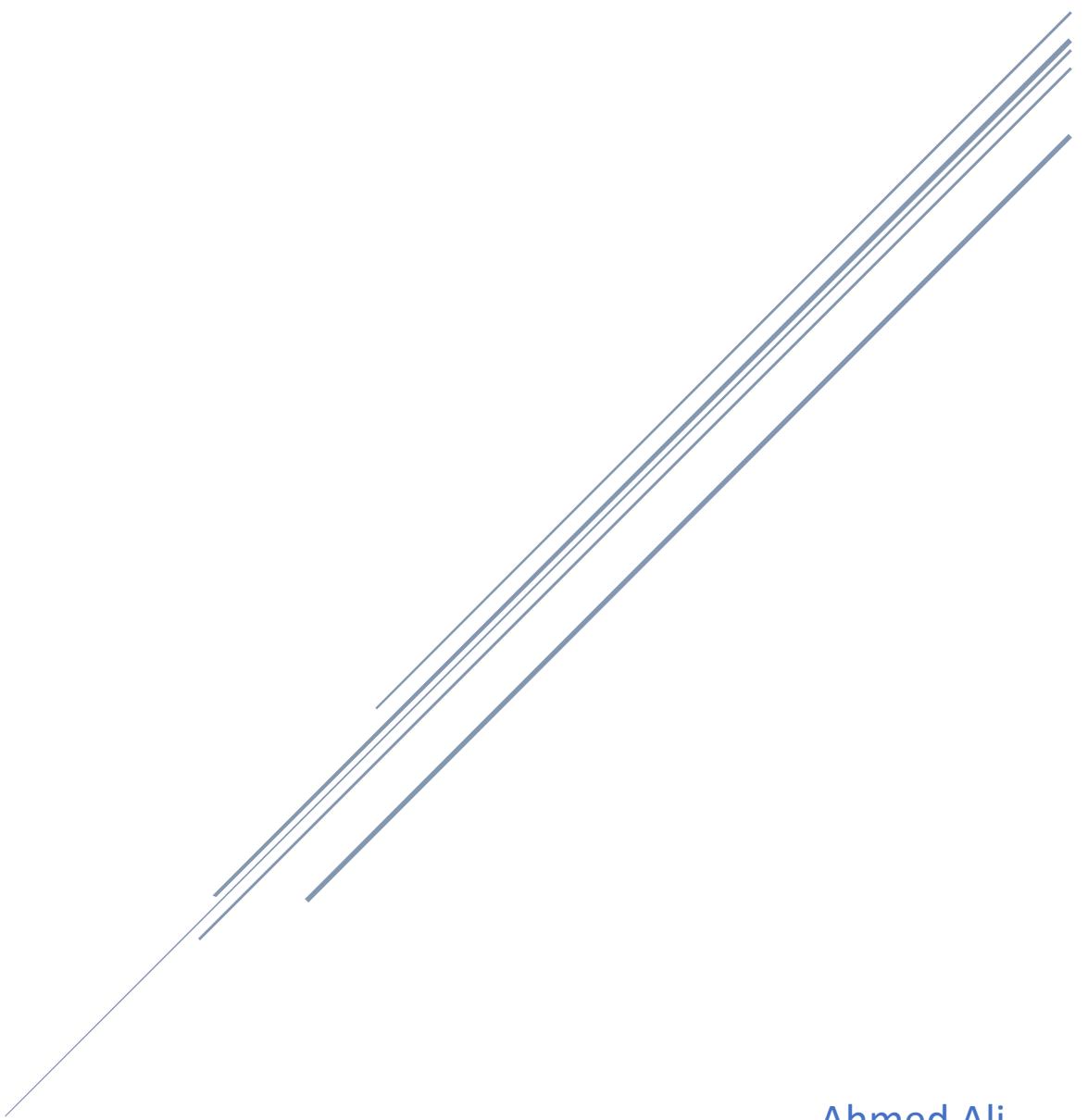


OPERATING SYSTEM

Worksheet 3 (PintOS)



Ahmed Ali
S2201026

CONTENTS

- 1. Getting started**
- 2. Editing Code**
- 3. Running – Result**

1. Download and build a local version of Pintos OS

Fork a version of Pintos into **your own Gitlab workspace** and then clone the resulting repo locally.

The screenshot shows a GitLab project page for 'Pintos' (Project ID: 20793). A success message indicates the project was successfully forked. The repository has 11 commits, 1 branch, 0 tags, and 379 KiB of project storage. The description states it is an UWE variant of the Stanford Pintos Operating System. A commit log entry shows 'Update README.md' by Shancang Li 3 years ago, with a commit hash d05862eb. The master branch is selected. A note says the project is up-to-date with the upstream repository. Navigation links include README, LICENSE, Add CHANGELOG, Add CONTRIBUTING, Add Wiki, and Configure Integrations. A file list table shows 'devices' and 'examples' with their last commits and update times.

Name	Last commit	Last update
devices	Base Pintos implementation	7 years ago
examples	Base Pintos implementation	7 years ago

```
ahmed@ahmed-Virtual-Machine: /  
ahmed@ahmed-Virtual-Machine:$ git clone https://gitlab.uwe.ac.uk/Ahmed18.Ali/Pintos.git  
fatal: could not create work tree dir 'Pintos': Permission denied  
ahmed@ahmed-Virtual-Machine:$ sudo git clone https://gitlab.uwe.ac.uk/Ahmed18.Ali/Pintos.git  
[sudo] password for ahmed:  
Cloning into 'Pintos'...  
Username for 'https://gitlab.uwe.ac.uk': a235-ali  
Password for 'https://a235-ali@gitlab.uwe.ac.uk':  
remote: Enumerating objects: 662, done.  
remote: Counting objects: 100% (662/662), done.  
remote: Compressing objects: 100% (561/561), done.  
remote: Total 662 (delta 93), reused 662 (delta 93), pack-reused 0  
Receiving objects: 100% (662/662), 357.64 KiB | 487.00 KiB/s, done.  
Resolving deltas: 100% (93/93), done.  
Checking connectivity... done.  
ahmed@ahmed-Virtual-Machine:$
```

- a. To build Pintos you must run make run different directories. First, cd into the **threads** directory and then issue the **make** command.

```
ahmed@ahmed-Virtual-Machine: /Pintos/threads  
threads/thread.o threads/switch.o threads/interrupt.o threads/intr-stubs.o thre  
ads/synch.o threads/palloc.o threads/malloc.o devices/pit.o devices/timer.o dev  
ices/kbd.o devices/vga.o devices/serial.o devices/block.o devices/partition.o dev  
ices/ide.o devices/input.o devices/intq.o devices/rtc.o devices/shutdown.o devic  
es/speaker.o lib/debug.o lib/random.o lib/stdio.o lib/stdcblib.o lib/string.o lib  
/arithmetic.o lib/ustar.o lib/kernel/debug.o lib/kernel/list.o lib/kernel/bitmap.  
o lib/kernel/hash.o lib/kernel/console.o tests/threads/tests.o tests/threads/ala  
rm-wait.o tests/threads/alarm-simultaneous.o tests/threads/alarm-priority.o test  
s/threads/alarm-zero.o tests/threads/alarm-negative.o tests/threads/priority-cha  
nge.o tests/threads/priority-donate-one.o tests/threads/priority-donate-multiple  
.o tests/threads/priority-donate-multiple2.o tests/threads/priority-donate-nest.  
.o tests/threads/priority-donate-sema.o tests/threads/priority-donate-lower.o tes  
ts/threads/priority-fifo.o tests/threads/priority-preempt.o tests/threads/pri  
ority-sema.o tests/threads/priority-condvar.o tests/threads/priority-donate-chain.o  
tests/threads/mlfqss-load-1.o tests/threads/mlfqss-load-60.o tests/threads/mlfqss-  
load-avg.o tests/threads/mlfqss-recent-1.o tests/threads/mlfqss-fair.o tests/threa  
ds/mlfqss-block.o  
objcopy -R .note -R .comment -S kernel.o kernel.bin  
gcc -m32 -c ../../threads/loader.S -o threads/loader.o -Wa,--gstabs -nostdinc -I  
../../ -I../../lib  
ld -melf_i386 -N -e 0 -Ttext 0x7c00 --oformat binary -o loader.bin threads/loade  
r.o  
make[1]: Leaving directory '/Pintos/threads/build'  
ahmed@ahmed-Virtual-Machine:/Pintos/threads$
```

Run ‘make’ to compile examples

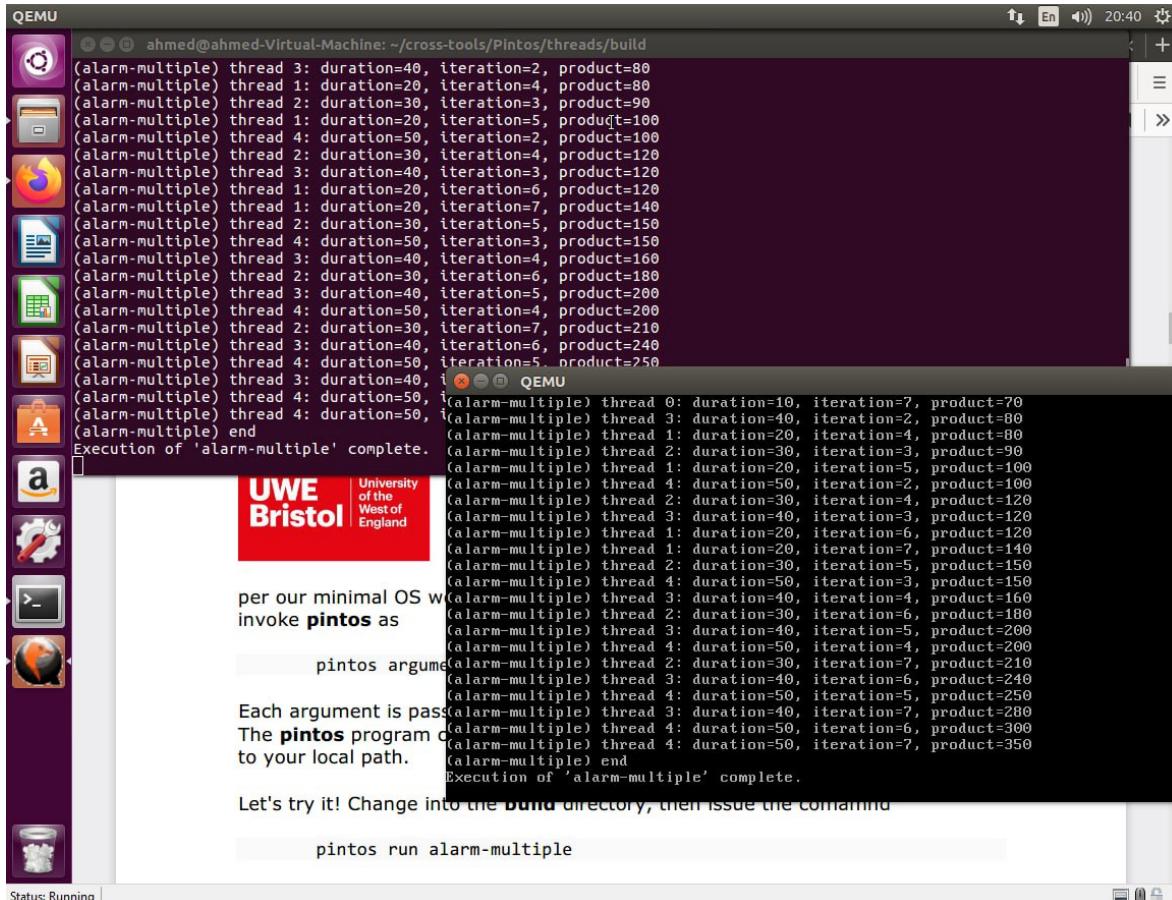
RESULT

```
ahmed@ahmed-Virtual-Machine: /Pintos/examples
types -Wmissing-prototypes -Wsystem-headers -MMD -MF insult.d
gcc -m32 -Wl,--build-id=none -nostdlib -static -Wl,-T,../lib/user/user.lds insult.o lib/user/entry.o libc.a -o insult
gcc -m32 -c lineup.c -o lineup.o -g -fno-strict-prototypes -Wmissing-prototypes -Wsystem-headers -MMD -MF lineup.d
gcc -m32 -Wl,--build-id=none -nostdlib -static -Wl,-T,../lib/user/user.lds lineup.o lib/user/entry.o libc.a -o lineup
gcc -m32 -c matmult.c -o matmult.o -g -fno-strict-prototypes -Wmissing-prototypes -Wsystem-headers -MMD -MF matmult.d
gcc -m32 -Wl,--build-id=none -nostdlib -static -Wl,-T,../lib/user/user.lds matmult.o lib/user/entry.o libc.a -o matmult
gcc -m32 -c recursor.c -o recursor.o -g -fno-strict-prototypes -Wmissing-prototypes -Wsystem-headers -MMD -MF recursor.d
gcc -m32 -Wl,--build-id=none -nostdlib -static -Wl,-T,../lib/user/user.lds recursor.o lib/user/entry.o libc.a -o recursor
gcc -m32 -c my.c -o my.o -g -fno-strict-prototypes -Wmissing-prototypes -Wsystem-headers -MMD -MF my.d
gcc -m32 -Wl,--build-id=none -nostdlib -static -Wl,-T,../lib/user/user.lds my.o lib/user/entry.o libc.a -o my
ahmed@ahmed-Virtual-Machine:/Pintos/examples$
```

Run ‘make’ to compile user interface programs. Result as follows:

```
ahmed@ahmed-Virtual-Machine: /Pintos/userprog
lib/user/entry.o libc.a -o tests/filesys/base/syn-remove
gcc -m32 -c ../../tests/filesys/base/syn-write.c -o tests/filesys/base/syn-write.o -g -fno-strict-protector -fno-stack-protector -nostdinc -I../../lib -I../../lib/user -I. -Wall -W -Wstrict-prototypes -Wmissing-prototypes -Wsystem-headers -MMD -MF tests/filesys/base/syn-write.d
gcc -m32 -Wl,--build-id=none -nostdlib -static -Wl,-T,../lib/user/user.lds tests/filesys/base/syn-write.o tests/lib.o tests/filesys/seq-test.o tests/main.o lib/user/entry.o libc.a -o tests/filesys/base/syn-write
gcc -m32 -c ../../tests/filesys/base/child-syn-read.c -o tests/filesys/base/child-syn-read.o -g -fno-strict-prototypes -fno-stack-protector -nostdinc -I../../lib -I../../lib/user -I. -Wall -W -Wstrict-prototypes -Wmissing-prototypes -Wsystem-headers -MMD -MF tests/filesys/base/child-syn-read.d
gcc -m32 -Wl,--build-id=none -nostdlib -static -Wl,-T,../lib/user/user.lds tests/filesys/base/child-syn-read.o tests/lib.o tests/filesys/seq-test.o lib/user/entry.o libc.a -o tests/filesys/base/child-syn-read
gcc -m32 -c ../../tests/filesys/base/child-syn-wrt.c -o tests/filesys/base/child-syn-wrt.o -g -fno-strict-prototypes -fno-stack-protector -nostdinc -I../../lib -I../../lib/user -I. -Wall -W -Wstrict-prototypes -Wmissing-prototypes -Wsystem-headers -MMD -MF tests/filesys/base/child-syn-wrt.d
gcc -m32 -Wl,--build-id=none -nostdlib -static -Wl,-T,../lib/user/user.lds tests/filesys/base/child-syn-wrt.o tests/lib.o tests/filesys/seq-test.o lib/user/entry.o libc.a -o tests/filesys/base/child-syn-wrt
make[1]: Leaving directory '/Pintos/userprog/build'
ahmed@ahmed-Virtual-Machine:/Pintos/userprog$
```

Running Pintos



```
ahmed@ahmed-Virtual-Machine: ~/cross-tools/Pintos/threads/build
Specify the 'raw' format explicitly to remove the restrictions.
PiLo hda1
Loading.....
Kernel command line: -h

Command line syntax: [OPTION...] [ACTION...]
Options must precede actions.
Actions are executed in the order specified.

Available actions:
  run TEST          Run TEST.

Options:
  -h                Print this help message and power off.
  -q                Power off VM after actions or on panic.
  -r                Reboot after actions.
  -rs=SEED          Set random number seed to SEED.
  -mlfq            Use multi-level feedback queue scheduler.

Timer: 0 ticks
Thread: 0 idle ticks, 0 kernel ticks, 0 user ticks
Console: 552 characters output
Keyboard: 0 keys pressed
Powering off...
ahmed@ahmed-Virtual-Machine:~/cross-tools/Pintos/threads/build$
```

USING THE FILE SYSTEM

create a simulated disk with a file system partition

```
ahmed@ahmed-Virtual-Machine: ~/cross-tools/Pintos/userprog/build$ pintos-mkdisk filesys.dsk --filesys-size=2
ahmed@ahmed-Virtual-Machine:~/cross-tools/Pintos/userprog/build$ pintos -f -q
Prototype mismatch: sub main::SIGVTALRM () vs none at /home/ahmed/cross-tools/Pintos/utils/pintos line 940.
Constant subroutine SIGVTALRM redefined at /home/ahmed/cross-tools/Pintos/utils/pintos line 932.
qemu-system-i386 -device isa-debug-exit -hda /tmp/7_Vlrv2M_q.dsk -hdb filesys.dsk -m 4 -net none -serial stdio
WARNING: Image format was not specified for '/tmp/7_Vlrv2M_q.dsk' and probing guessed raw.
        Automatically detecting the format is dangerous for raw images, write operations on block 0 will be restricted.
        Specify the 'raw' format explicitly to remove the restrictions.
WARNING: Image format was not specified for 'filesys.dsk' and probing guessed raw.
        Automatically detecting the format is dangerous for raw images, write operations on block 0 will be restricted.
        Specify the 'raw' format explicitly to remove the restrictions.
PiLo hda1
Loading.....
Kernel command line: -f -q
Pintos booting with 3,968 kB RAM...
367 pages available in kernel pool.
367 pages available in user pool.
Calibrating timer... 97,280,000 loops/s.
hda: 1,008 sectors (504 kB), model "QM00001", serial "QEMU HARDDISK"
hda1: 182 sectors (91 kB), Pintos OS kernel (20)
hdb: 5,040 sectors (2 MB), model "QM00002", serial "QEMU HARDDISK"
hdb1: 4,096 sectors (2 MB), Pintos file system (21)
filesys: using hdb1
Formatting file system...done.
Boot complete.
Timer: 64 ticks
Thread: 0 idle ticks, 64 kernel ticks, 0 user ticks
hdb1 (filesys): 3 reads, 6 writes
Console: 580 characters output
Keyboard: 0 keys pressed
Exception: 0 page faults
Powering off...
ahmed@ahmed-Virtual-Machine:~/cross-tools/Pintos/userprog/build$
```

Then format the file system partition as shown above giving those results

```
ahmed@ahmed-Virtual-Machine:~/cross-tools/Pintos/userprog/build$ ls
devices  filesys.dsk  kernel.o  libc.a    Makefile  threads
filesys  kernel.bin  lib       loader.bin  tests    userprog
ahmed@ahmed-Virtual-Machine:~/cross-tools/Pintos/userprog/build$
```

After filesys.dsk creation you need to run pintos -f -q and the following should be seen.

```
ahmed@ahmed-Virtual-Machine: ~/cross-tools/Pintos/userprog/build
W.
        Automatically detecting the format is dangerous for raw images, write o
perations on block 0 will be restricted.
        Specify the 'raw' format explicitly to remove the restrictions.
PiLo hd1
Loading.....
Kernel command line: -f -q
Pintos booting with 3,968 kB RAM...
367 pages available in kernel pool.
367 pages available in user pool.
Calibrating timer... 389,120,000 loops/s.
hda: 1,008 sectors (504 kB), model "QM00001", serial "QEMU HARDDISK"
hda1: 182 sectors (91 kB), Pintos OS kernel (20)
hdb: 5,040 sectors (2 MB), model "QM00002", serial "QEMU HARDDISK"
hdb1: 4,096 sectors (2 MB), Pintos file system (21)
filesys: using hdb1
Formatting file system...done.
Boot complete.
Timer: 67 ticks
Thread: 0 idle ticks, 67 kernel ticks, 0 user ticks
hdb1 (filesys): 3 reads, 6 writes
Console: 581 characters output
Keyboard: 0 keys pressed
Exception: 0 page faults
Powering off...
ahmed@ahmed-Virtual-Machine:~/cross-tools/Pintos/userprog/build$
```

```
ahmed@ahmed-Virtual-Machine: ~/cross-tools/Pintos/userprog/build
Loading.....
Kernel command line: -q extract
Pintos booting with 3,968 kB RAM...
367 pages available in kernel pool.
367 pages available in user pool.
Calibrating timer... 625,049,600 loops/s.
hda: 1,008 sectors (504 kB), model "QM00001", serial "QEMU HARDDISK"
hda1: 182 sectors (91 kB), Pintos OS kernel (20)
hda2: 76 sectors (38 kB), Pintos scratch (22)
hdb: 5,040 sectors (2 MB), model "QM00002", serial "QEMU HARDDISK"
hdb1: 4,096 sectors (2 MB), Pintos file system (21)
filesys: using hdb1
scratch: using hda2
Boot complete.
Extracting ustar archive from scratch device into file system...
Putting 'echo' into the file system...
Erasing ustar archive...
Timer: 80 ticks
Thread: 3 idle ticks, 77 kernel ticks, 0 user ticks
hdb1 (filesys): 25 reads, 150 writes
hda2 (scratch): 75 reads, 2 writes
Console: 788 characters output
Keyboard: 0 keys pressed
Exception: 0 page faults
Powering off...
ahmed@ahmed-Virtual-Machine:~/cross-tools/Pintos/userprog/build$
```

```
ahmed@ahmed-Virtual-Machine: ~/cross-tools/Pintos/userprog/build
    Automatically detecting the format is dangerous for raw images, write o
perations on block 0 will be restricted.
        Specify the 'raw' format explicitly to remove the restrictions.
WARNING: Image format was not specified for 'filesys.dsk' and probing guessed ra
w.
    Automatically detecting the format is dangerous for raw images, write o
perations on block 0 will be restricted.
        Specify the 'raw' format explicitly to remove the restrictions.
PiLo hda1
Loading.....
Kernel command line: -q run 'echo x'
Pintos booting with 3,968 kB RAM...
367 pages available in kernel pool.
367 pages available in user pool.
Calibrating timer... 389,120,000 loops/s.
hda: 1,008 sectors (504 kB), model "QM00001", serial "QEMU HARDDISK"
hda1: 182 sectors (91 kB), Pintos OS kernel (20)
hdb: 5,040 sectors (2 MB), model "QM00002", serial "QEMU HARDDISK"
hdb1: 4,096 sectors (2 MB), Pintos file system (21)
filesys: using hdb1
Boot complete.
Executing 'echo x':
load: echo x: open failed
```

```
pintos -q run 'echo x'
```

```
QEMU
SeaBIOS (version Ubuntu-1.8.2-1ubuntu1)
Booting from Hard Disk...
PiLo hda1
Loading.....
Kernel command line: -q run 'echo x'
Pintos booting with 3,968 kB RAM...
367 pages available in kernel pool.
367 pages available in user pool.
Calibrating timer... 389,120,000 loops/s.
hda: 1,008 sectors (504 kB), model "QM00001", serial "QEMU HARDDISK"
hda1: 182 sectors (91 kB), Pintos OS kernel (20)
hdb: 5,040 sectors (2 MB), model "QM00002", serial "QEMU HARDDISK"
hdb1: 4,096 sectors (2 MB), Pintos file system (21)
filesys: using hdb1
Boot complete.
Executing 'echo x':
load: echo x: open failed
```

EDITING CODE :

```
ahmed@ahmed-Virtual-Machine: ~/cross-tools/Pintos/userprog/build
on block 0 will be restricted.
    Specify the 'raw' format explicitly to remove the restrictions.
Pilo hda1
Loading.....
Kernel command line: -q run echo
Pintos booting with 3,968 kB RAM...
367 pages available in kernel pool.
367 pages available in user pool.
Calibrating timer... 419,020,800 loops/s.
hda: 1,008 sectors (504 kB), model "QM00001", serial "QEMU HARDDISK"
hda1: 182 sectors (91 kB), Pintos OS kernel (20)
hdb: 5,040 sectors (2 MB), model "QM00002", serial "QEMU HARDDISK"
hdb1: 4,096 sectors (2 MB), Pintos file system (21)
filesys: using hdb1
Boot complete.
Executing 'echo':
Page fault at 0xc0000008: rights violation error reading page in user context.
echo: dying due to interrupt 0x0e (#PF Page-Fault Exception).
Interrupt 0x0e (#PF Page-Fault Exception) at eip=0x80480f1
cr2=c0000008 error=00000005
eax=00000000 ebx=00000000 ecx=00000000 edx=00000000
esi=00000000 edi=00000000 esp=bfffffec ebp=00000000
cs=001b ds=0023 es=0023 ss=0023
ahmed@ahmed-Virtual-Machine:~/cross-tools/Pintos/userprog/build$
```

Now when this page fault occurs: Change `*esp = PHYS_BASE;`

to `*esp = PHYS_BASE - 12;`

```
ahmed@ahmed-Virtual-Machine: ~/cross-tools/Pintos/userprog
GNU nano 2.5.3                               File: process.c                         Modified

kpage = palloc_get_page (PAL_USER | PAL_ZERO);
if (kpage != NULL)
{
    success = install_page (((uint8_t *) PHYS_BASE) - PGSIZE, kpage, true);
    if (success) {
        *esp = PHYS_BASE - 12;
    } else
        palloc_free_page (kpage);
}
return success;
}

/* Adds a mapping from user virtual address UPAGE to kernel
   virtual address KPAGE to the page table.
   If WRITABLE is true, the user process may modify the page;
   otherwise, it is read-only.
   UPAGE must not already be mapped.
   KPAGE should probably be a page obtained from the user pool
   with palloc_get_page(). */

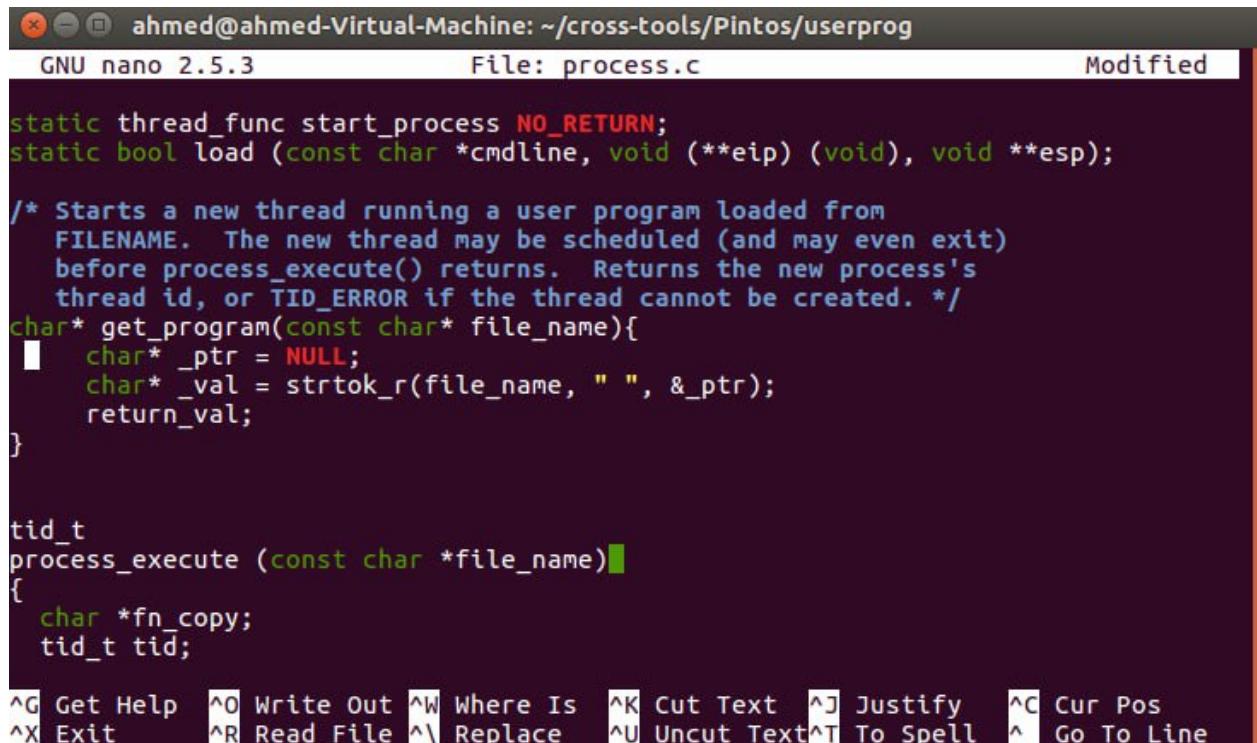
^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell  ^  Go To Line
```

User program .

We have to set the path to

```
``  
export PREFIX="$HOME/cross-tools"  
export TARGET=i686-elf  
export PATH="$PREFIX/bin:$PATH"  
  
export PATH="$PATH:$HOME/cross-tools/Pintos/utils/"
```

Create a new function to extract the program name from an argument string that contains the program name and any number of command-line parameters.



The screenshot shows a terminal window titled "ahmed@ahmed-Virtual-Machine: ~/cross-tools/Pintos/userprog". The file being edited is "process.c". The code in the editor is:

```
static thread_func start_process NO_RETURN;  
static bool load (const char *cmdline, void (**eip) (void), void **esp);  
  
/* Starts a new thread running a user program loaded from  
 FILENAME. The new thread may be scheduled (and may even exit)  
 before process_execute() returns. Returns the new process's  
 thread id, or TID_ERROR if the thread cannot be created. */  
char* get_program(const char* file_name){  
    char* _ptr = NULL;  
    char* _val = strtok_r(file_name, " ", &_ptr);  
    return _val;  
}  
  
tid_t  
process_execute (const char *file_name)  
{  
    char *fn_copy;  
    tid_t tid;
```

At the bottom of the screen, there is a menu bar with various keyboard shortcuts for the nano editor, including:

- ^G Get Help
- ^O Write Out
- ^W Where Is
- ^K Cut Text
- ^J Justify
- ^C Cur Pos
- ^X Exit
- ^R Read File
- ^V Replace
- ^U Uncut Text
- ^T To Spell
- ^ Go To Line

Start process to ..

```
ahmed@ahmed-Virtual-Machine: ~/cross-tools/Pintos/userprog
GNU nano 2.5.3           File: process.c          Modified |
```

```
/* A thread function that loads a user process and starts it
running. */
static void
start_process (void *file_name_)
{
    char *file_name = file_name_;
    struct intr_frame if_;
    bool success;

    /* Initialize interrupt frame and load executable. */
    memset (&if_, 0, sizeof if_);
    if_.gs = if_.fs = if_.es = if_.ds = if_.ss = SEL_UDSEG;
    if_.cs = SEL_UCSEG;
    if_.eflags = FLAG_IF | FLAG_MBS;

    //get program name from argument
    char* _name = get_program(file_name);■

    success = load (file_name, &if_.eip, &if_.esp);
    /* If load failed, quit. */
    palloc_free_page (file_name);
    if (!success)
        thread_exit ();

    /* Start the user process by simulating a return from an
    interrupt, implemented by intr_exit (in
    threads/intr-stubs.S). Because intr_exit takes all of its
    arguments on the stack in the form of a `struct intr_frame',
    we just point the stack pointer (%esp) to our stack frame
    and jump to it. */
    asm volatile ("movl %0, %%esp; jmp intr_exit" : : "g" (&if_) : "memory");
    NOT_REACHED ();
}

/* Waits for thread TID to die and returns its exit status. If
```

^G Get Help **^O** Write Out **^W** Where Is **^K** Cut Text **^J** Justify **^C** Cur Pos
^X Exit **^R** Read File **^V** Replace **^U** Uncut Text **^T** To Spell **^L** Go To Line

To implement exit code add to process_exit ..

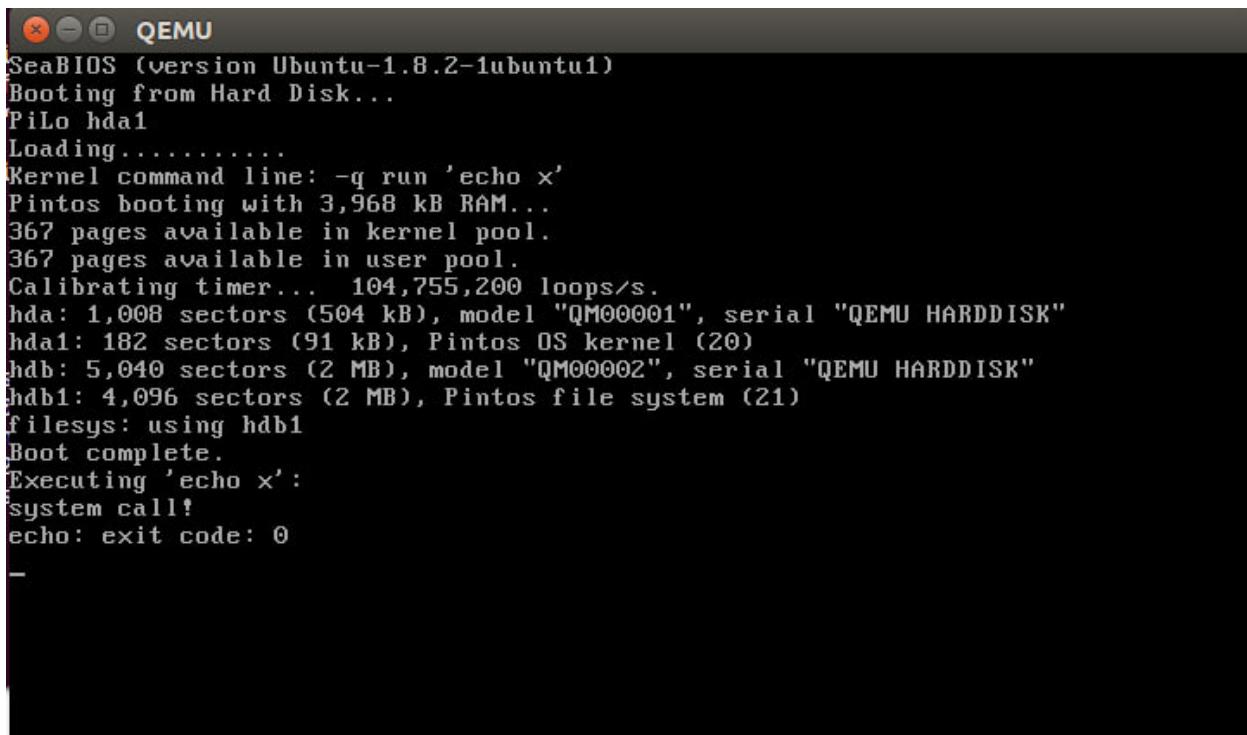
```
ahmed@ahmed-Virtual-Machine: ~/cross-tools/Pintos/userprog
GNU nano 2.5.3          File: process.c          Modified |
```

```
{  
    struct thread *cur = thread_current ();  
    uint32_t *pd;  
  
    /* Destroy the current process's page directory and switch back  
       to the kernel-only page directory. */  
    pd = cur->pagedir;  
    if (pd != NULL)  
    {  
        /* Correct ordering here is crucial. We must set  
           cur->pagedir to NULL before switching page directories,  
           so that a timer interrupt can't switch back to the  
           process page directory. We must activate the base page  
           directory before destroying the process's page  
           directory, or our active page directory will be one  
           that's been freed (and cleared). */  
        cur->pagedir = NULL;  
        pagedir_activate (NULL);  
        pagedir_destroy (pd);  
    }  
  
    //print EXIT_CODE  
    printf ("%s: exit code: %d\n", cur -> name, cur -> exitcode);  
  
}  
  
/* Sets up the CPU for running user code in the current  
   thread.  
   This function is called on every context switch. */  
void  
process_activate (void)  
{  
    struct thread *t = thread_current ();  
  
    /* Activate thread's page tables. */  
    pagedir_activate (t->pagedir);  
  
    /* Set thread's kernel stack for use in processing  
       interrupts. */  
    tss_update ();  
  
    ^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos  
    ^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell  ^_ Go To Line
```

And to thread.h

```
#ifdef USERPROG  
    /* Owned by userprog/process.c. */  
  
    int8_t* exitcode; // EXIT_CODE  
  
    uint32_t *pagedir;           /* Page directory. */
```

Running – Result



The screenshot shows a terminal window titled "QEMU" running SeaBIOS (version Ubuntu-1.8.2-1ubuntu1). The window displays the boot process, including the loading of the PLo hda1 kernel, memory allocation, timer calibration, and the booting of the Pintos OS kernel from hda1. It also shows the execution of a user command "echo x" via a system call, resulting in an exit code of 0.

```
SeaBIOS (version Ubuntu-1.8.2-1ubuntu1)
Booting from Hard Disk...
PiLo hda1
Loading.....
Kernel command line: -q run 'echo x'
Pintos booting with 3,968 kB RAM...
367 pages available in kernel pool.
367 pages available in user pool.
Calibrating timer... 104,755,200 loops/s.
hda: 1,008 sectors (504 kB), model "QM00001", serial "QEMU HARDDISK"
hda1: 182 sectors (91 kB), Pintos OS kernel (20)
hdb: 5,040 sectors (2 MB), model "QM00002", serial "QEMU HARDDISK"
hdb1: 4,096 sectors (2 MB), Pintos file system (21)
filesys: using hdb1
Boot complete.
Executing 'echo x':
system call!
echo: exit code: 0
-
```