ASSIGNMENT-2

NAME-SHISHU

R&G-2020CA0089

(1) Write your xv6 installation experience.

My xv6 installation experience is very good. but I found some problem during the installation of all packages which is required for xv6.

At begning my Internet connection is not good so I an anable to install any package but where I fix it the all things works properly.

First of all I update my system using "sudo opt-get update" command after that I upgrade my system.

After that I am installing build-essentials which is a package containing important applications for developers. for cross compilation gcc-multilib which is used to compile a program for a 32 bit artitecture in 64-bit machine.

After that I am installing qemu and git-core on installing qemu I found some dificulties on installing bridge-utils. But on installing aptitude I get success.

At last I doconload xv6 using the comand.

"git clone git://github.com/mit-pdos/xv6-public.git.

After that I use some comands:- cd xv6
make, make qemu, ls.

(2) Write about the xv6 operating system in 1000 words

XV6 is a ~~machine~~ morden reimplementation of Sixth Edition Unix in ANSI for multiprocessor x86 and RISC-V system. It was created for pedagogical purpose in MIT's operating system Engineering course. It has following new properties which make it different from original.

* A yield system call is added.
* A shutdown utility and system call is added (only ~~cor~~ in qemu).
* The default number of cores to run with is changed to 1.
* User programs are linked against Ub gcc a, so things like long long works, this means however that some modifications will be needed to build this version of xv6 with along or other compilers.
* The V2P, P2V, PTE-ADDR, and PTE_FLAGS macros are changed into static inline functions which provide slightly better error checking.
* The makefile is modified so assembly files depend on their #include files.
* The FSSIZE value in param.h was increased.
* The Makefile was modifide to delete intermediate files, so a failed mkfs does not leave behind a filesystem image.
* mkfs was modified to give more consistent and information errors when it runs out of space in the disk image.

* A makefile target to add in submission is added.

* The linker script is updated to avoid a problem with recent version of binutil that prevented XV6 from booting with some recent Linux version (including Arach Linux).

Files in XV6 :-

To help navigate the XV6 source code a brief explanation of some selected files :-

* shared user/kernal header and utility files. types.h, fentl.h, stat.h.

* Utitiy (non-xv6) programs.
  mkfs.c - create filesystem images - to xv6 can boot in qemu.

* User mode code -
  user.h - declarations of system calls wrappers and standerd library function
  usys.h = assembly code (generated by preproccessor macros) for system call wrapers. ulib.c, printf.c, malloc.c - user standerd library, including printf, malloc free.

  It supplies xv6 program like cat.c echo.c, forktest.c, grep.c, init.c etc.

* Kernel mode code everything else
  → defs.h - declarations of functions callable within the kernal.
  → process - related :-
  proc.h, proc.c
  exec.c, elf.h - loading executables into memory. The BSD 2 format also MB on the host. The allows the creation of overlay images that

memory.

file.h, file.c, pipe.c - file descripter handling related code

→ memory management
mmu.h, vm.c, kalloc.h

→ multicore -
mp.c, mp.h

→ system call handling
syscall.h, syscall.c
sysproc.c, sysfile.c

→ I/O
buf.h, bio.c, console.c, ide.c

→ file system :-
fs.h, fs.c, log.c

→ boot handling :-
bootmain.c, bootasm.s, main.c

(3) Write about the "qemu" in 1500 words

(Quik EMUlator) open source software for creating emulation and virtual machine environments, developed by Fabrice Bellard. As an Emulater it is used to run operating system and applications written for another hardware platform; for Example running ARM software on an x86-based PC. for virtulizat-on QEMU is used to emulate devices and certain privilleged instructions and requires either the KQEMU or KVM kernal modes and the host operating system to provide a virtual machine envi-ronment. It is typically used to run windows and DOS applications on x86 based Linux on computer. It is free and open source by pervisor. it emulates the machine processor through dynamic

binary translation and provides a set of different hardware and device models for the machine. enabling it to run a variety of guest operating systems. It can interoperate with kernel based virtual machine (KVM) to run virtual machines at near-native speed. QEMU can also do emulation for user-level processes, allowing applications compiled for one architecture to run another.

Operating modes :-

* User-mode emulation :- In this mode qemu runs single Linux or Darwin/mac programs that were compiled for a different instruction set. System calls are thunked for endianess and for 32/64 bit machine mismatches. fact cross compilation and cross-debugging are the main targets for user mode emulations.

* System emulation :- In this mode qemu emulates a full computer system, including peripherals. It can be used to provide virtual hosting of several virtual computers on a single computer. qemu can boot many guest operating systems, including Linux., Solaris, Microsoft, windows, DOS and BSD. It supports emulating several instruction sets including x86, MIPS, 32-bit ARMV8, PowerPC, MicroBlaze etc.

* KVM hosting :- Here qemu deals with the setting up and migration of KVM images. It is still involved in the emulation of hardware, but the execution of the

guest is done by KVM as requested by qemu.

* Xen Hosting :- qemu is involved only in the emulation of hardware; the execution of the guest is done within Xen and is totally hidden from qemu.

Features :-

QEMU can save and restore the state of the virtual machine with all programs running. Guest operating systems do not need patching in order to run inside QEMU.

QEMU supports the emulation of various architectures, including x86, MIPS 64 (up to related 6) SPARC (sun4m, sun4a), ARM (Integrator/cp and versatile/PB), SuperH, PowerPC (PReP and Power Macintosh), ETRAX CRIS, MicroBlaze, and RISC-V.

Virtual Disk images can be stored in a special format (qcow, qcow2) that only takes up as much disk-space as the guest OS actually uses. This way an emulated 120 gb disk may occur only a few hundred MB on the host. The QCOW2 format also allows the ~~certain~~ creation of overlay images that record the difference from another (unmodified) base image file. This provide the possibility for reverting the emulated disk's content to an earlier state. for Example a base images are

used should the guest system become unusable (through virus attack, accidental system distruction etc). the user can delete the overlay and use an earlier emulated disk image.

QEMU integrate several services to allow the host and guest systems to communicate for Example, and integrated SMB server and network port redirection (allow incoming connection to the Virtual machine). It can also boot Linux kernals without a bootloader.

Simulating multiple CPU's running SMP is possible.

QEMU does not requires administrative rights to run unless additional kernal modules for improving speed (like KQEMU) are used or certain modes of its network connectivity model are utilized.