VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

NÁZEV PRÁCE

THESIS TITLE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JMÉNO PŘÍJMENÍ**
AUTHOR

VEDOUCÍ PRÁCE **Doc. RNDr. JMÉNO PŘÍJMENÍ, Ph.D.**
SUPERVISOR

BRNO 2018

Abstrakt

Do tohoto odstavce bude zapsán výtah (abstrakt) práce v českém (slovenském) jazyce.

Abstract

Do tohoto odstavce bude zapsán výtah (abstrakt) práce v anglickém jazyce.

Klíčová slova

Sem budou zapsána jednotlivá klíčová slova v českém (slovenském) jazyce, oddělená čárkami.

Keywords

Sem budou zapsána jednotlivá klíčová slova v anglickém jazyce, oddělená čárkami.

Citace

PŘÍJMENÍ, Jméno. *Název práce*. Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Doc. RNDr. Jméno Příjmení, Ph.D.

Název práce

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana X... Další informace mi poskytli... Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

....
Jméno Příjmení
25. února 2018

Poděkování

V této sekci je možno uvést poděkování vedoucímu práce a těm, kteří poskytli odbornou pomoc (externí zadavatel, konzultant, apod.).

Obsah

Kapitola 1

Virtualization

In first chapter I will explain what virtualiszation is and how it can be implemented.

- Virtualization basics,
- Advantages,
- Disadvantages,
- Virtualization architectures.

1.1 Virtualization basics

In short, virtualization means emulation of hardware within a software platform. Although it may seem that virtuaization is the invention of last few years its concept was first brought up in 1960s. It was first implemented by IBM to split resources of their massive mainframe machines among multiple virtual machines that could work independently of each other and therefore use resources more efficiently.

To furher explain basics of virtualization I would like to start by brief explanation of architecture of standard computers. First, we need hardware layer which contains processing unit, storage and other hardware devices, on top of this layer is operating system which abstracts functioning of hardware layer and offers application layer medium to communicate with HW layer. Operating system runs as privilaged software which means that is generally able to perform any operation supported by hardware, its role is to create interface which simplifyes or ingores implemnentation of components on lower levels of a hierarchy to make creating and using components on application level easier. On top of operating system is application layer which consists of user programs whis are less privileged and can generally perform only operations that are permitted to them by operating system. [obrazok]

Virtualization allows creating multiple entities of os and application layers on single hw layer. This is done by inserting additional layer of system software between operating system and lower layers. This virtualization layer is called hypervisor or VMM (virtual machine monitor) and there are two main types of it:

- Type I hypervisor (or bare-metal/native hypervispor): runs directly on host system's hardware with highes level of privilege and has full control over application layer running on top of it. Type I hypervisors have generally better performance than type II hypervisors because they don't have to communicate with hardware layer throught operating system thus can utilize full potencial it. Some of them need special priviledged virtual machine called Domain-0 from which it can be managed and controlled. [obrazok]

- Type II hypervisor (or hosted hypervisor): can be either on same level as host operating system or on a level above. This means that hypervisor doesn't require specific drivers for I/O operations and allows running of virtual environment within alredy existing environment. [obrazok popisujuci strukturu]

1.2 Advantages

As every other technology, virtualization have some advantages and disadvantages. Lets start with advantages:

1. Efficiency: Virtualization allows more efficient use of host machine for multiple virtual machines which can all run different services. Runnig multiple services on same server can be very dangerous multiple reasons and so it is considered a bad practice. Most obvious problem is with hardware overhead, if numerous very resource-demanding services run on single physical machine it can slow them down considerably or even crash them which makes business's services very unreliable. On the other hand, if we run only single service on whole physical host, that service can use for example only 20% of server machine's resources (which is especially nowdays very common as hardware is becomming more and more powerful, average service utilizes only about 10% to 15% of all available resources). This way we waste remaining 80% of resources which in the long unnecessarily raises expenses on managing physical hosts. Rather we can run various virtual servers on one physical hardware and on each of them run single service. Over past few years server technlogy has improved som much that wasting server's machine resources this way is very common and can by improved by virtualization.
2. Cost reducing: By concentrating virtual machines on fever hosts we can reduce expenses associated with running large numbers of physical machines in various ways. We need less physical space for storing hosts, less cooling is needed, less energy is spent on powering all the hardware, managing fewer physical devices is also easier and cheaper. Companies with fewer than 1000 employees spend up to 40% of their IT budget on hardware[], this can be greatly reduced by virtualization. Although this benefits more larger companies using plenty of server machines even smaller companies can benefit greatly from virtualization. Using less energy/space/etc means that virtualization is also firnedlier to environment.
3. Flexibility: Encreasing number of physical workstations or servers is financialy and time consuming process. We need new physical space, order new machines, set them up and so on. With virtual machines whole process is easier and faster. There are no more additional hardawe costs and administrators can easily setup and manage virtual machines using virtual machine management software. By using templates we can make creating new virtual machines even faster by automatization of setting up procedures. When hardware on which virtual machines are running becomes obsolete or it just needs to be out of service for maintenance resons we can easily migrate them to another physical hardware.

4. Testing: Virtual machines are completely isolated from each other which gives us possibility of testing environmnets with completely different operating systems and configurations. Even extreme situations are easy to set up and changed. Compared to physical machines, virtual machines can be added and removed very fast. QA teams often have multiple virtual machines with which they speed up testing and therefore development process.
5. Security: All virtual machines are isolated entities completely separated from every other software so when one of them gets attacked, gets virus or for any reason fails, only that one virtual machine fails and nothing other is affected. While problematic VM is diagnosed and repaired another VM can take it's place and continue running it's service which greatly reduces down-time and increases reliability of offered service.
6. Isolation: Virtual machines are hardware independent which means that their current state can be captured and reproduced on another physical host in process called migration. This reduces down-time even more becasue we can run all our services temporarily from another host while former host is being down due to maintenance. For example Red Hat Enterprise Virtualization supports live migration which is the ability to move running virtual machine between physical hosts with no interruption of service. The virtual machine remains powered on and user applications continue to run while the virtual machine is relocated to a new physical host. In the background, the virtual machine's RAM is copied from the source host to the the destination host. Storage and network connectivity are not altered. (citovane zo stranky red hatu)

1.3 Disadvantages

Disadvantages: The disagvantages of virtualization are mostly those that are associated with any transition to a new technology and can be overcame by careful planing and professional implementation.

1. Overloading: this problem lies in wrong or uncomplete estimation of amount of hardware resources needed to handle desired virtual environment. Virtualizaion carries along additional bandwidth in form of hypervisor and other components which is not neglectable. Other extreme is not utilizing full potencial of physical host capabilities and wasting resources in long run by using more hosts that are actually needed. Basic rule of thumb is to use around 80% of physical machines resources.
2. Bandwidth: the volume of data transfered through network might be noo much to handle for single network interface card (NIC), this can lead to slower network transfers. One of possible ways to solve this is to use host machine with multiple NICs.
3. Need for adjustments: in some cases, adapting a virtualization technology requires rewriting or patching some pieces of software to be compatible with virtual environment.
4. Cost: To run multiple machines on single host mahine we need it to be sufficiently powerful. This means that additional investments into hardware may be needed. Another investments are into virtualization software and managing virtual machines.
5. Learning curve: conversion to and managing virtual environment will require IT staff with necessary training. The beginning stages can be painful due to lack of experience with new technology.

6. Vulnerability: altough virtualization brings certain security benefits it also bring a big risk in form of potencial damage cost by lower level layers corruptoion. In physical environment if operating system of one machine gets infected than only that one machine is affected but in virtualized environment if hardware, operating system or hypervisor of host gets damaged than all virtual machines running on it can potencially become unavailable. This problem can be reduced by regular backups and snapshots which allow easy transfer of virtual machines to a new host.
7. Licencing: majority of software vendors consider virtual machine exactly the same as physical machine so if certain piece of software is needed on multiple virtual machines (for exmaple operating system) than we have to pay for a licence for each one of them. We can try to solve this by using open-source software. This is becoming less of a problem nowadays because more and more software vendors are adjusting their view on virtualization.

1.4 Virtualization architectures

Virtualization technology is spreading rapidly and today there are several architectures that implement this concept. Here are most used architectures and later we will discuss them in little more detail.

1. Full virtualization
2. Paravirtualization
3. Operating system virtualization
4. Other types of virtualization

1. Full virtualization: provides a total virtualization of hardware which means that every virtual entity runs as if was running on physical hardware completely unaware that its platform is virtualized. When a virtual machine wants to access hardware it access virtual hardware which access hypervisor which finally accesses physical hardware. Hypervisor thus acts as the only bridge between virtual machine and physical hardware. This is reason why full virtualization is in terms of performance behin non-virtualized machines.
   (a) Software assisted virtualization: [obrazok] historically first full virtualization solution introduced in 1969. This aproach has advantage in fact that if everything is simulated than any operting system or application can be run completly without modifications. On the other hand every operation made by operating system of virtual machine needs to be simulated and checked if it doesn't conflict with any other virtual machine or hypervisor which makes this process very resource-expensive.
   (b) Hardware assisted virtualization: [obrazok] introduced by IBM in 1972. This aproach reduces the problem of massive overhead in software assisted virtualization by extending functionality of hardware (mainly CPU) by new instructions allowing virtual machines to directly access physical hardware withou many expensive mediators. Normally x86 operating systems need to have a direct access to hardware resources, software-based virtualization solves this problem by virtualizing entire hardware layer but for price of wasting a big chunk of hardware resources. In hardware-base virtualization is this overhead noticibly reduced because processor no longer needs to be emulated and can directly interact with application layer.

Another advantage is that it will work 'right of the box' meaning that we don't neeed to upgrade or change anything, all we need to do is to use processor supporting hardware-based virtualization technology.

2. Paravirtualization: [k tomuto obrazok na hypercall] introduced in 1972 by IBM, this technique was implemented to increace performance of virtualized environmnet closer to non-virtualized. To use paravirtualization, kernel of the operating system needs to be modified, mainly it needs to have replaced any priviledged operation running only in ring 0 by so called 'hypercalls'. Hypercalls allow guest operating system to send system calls directly to he hypervisor without the need of hardware simulation. Virtual machine can therefore access some part of the hardware straight wihtout going throught virtualized hardware on top of hypervisor, which greatly increases performance of some operations. This only works for some parts of the hardware, for other parts, virtual machine still needs to access them via virtualized hardware. Paravirtualized virtual machine is 'aware' of the fact that it is being virtualized which gives it ability to use hypercalls. On the other hand, as mentioned above, in order to use this technique, operating system needs to be altered in a non-trivial way which typically limits its use to a Linux based operating systems because they allow such source code modifications. Paravirtualization was popularized by Xen hypervisor, today, most virtualization solutions use it as a norm (for example Microsoft HYper-V, Red Hat Xen, VMware's family and others).

3. Operating system virtualization: [obrazok] (also known as shared kernel virtualization) introduces 'light-weight' virtualization. To understant this concept, first we need at least very basic understandig of what are kernel and root file system and are their roles. Kernel is central part of the operating system, simply put, it mediates communication between operating system and physical hardware. Root file system contains all the files, libraries and practically all utilities necessary for operating system to work properly. In shared kernel virtualization every virtual machine has its own root file system but uses host operating system's kernel which they share among each other. Kernel has the ability to dynamically switch the current root file system to a different one without the need to reboot the whole system (technique known as 'chroot'). In this case, virtual machines are reffered to as 'containers' due to the fact that they are not completely separated but share host machine's kernel. Shared kernel means very little overhead compared to other virtualization concepts and thus high performance. Despite its light-weight nature, this concept also supports advanced features such as isolating memory space, regulating memory, network, CPU and I/O usage, some implementations even allow live migration. Biggest advantage of container-virtualization is superior efficiency and very little overhead compared to other types of virtualization because it doesn;t have to emulate all the hardware. Interactions between software and hardware are handled by operating systems kernel inside container. Major disadvantage of this technique is that container's operating system must be compatible with kernel on which it runs (container operating system must be designed for type and version of kernel that is being shared). Further, container environments cannot execute some top-level actions, mount/dismount file systems and so on whereas fully virtualized solutio gives us fully independent environment in which isn't user restricted in any way. Well known solutions are Linux VServer, Jail for FreeBSD, Zone for Solaris, Virtuozzo for Windows, Rosetta for Mac OS and others.

4. Other types of virtualization:

(a) Network virtualization: main idea is to create multiple virtual sub-networks (channels) that run on single physical network and are idependent from one another. Each one of virtual networks can have different bandwidth related, security or other restrictions and we can regulate trafic on each channel independently. Monitoring individual networks with their own purposes and settings is also easier and faster, it increases reliability and durability of network too as if one of virtual networks is for whatever reason overloaded or down other networks are not affected. Most modern hypervisors implement virtual networking in some form. Network virtualization can be further divided into two sub-categories:

   i. Internal: can be used for communication between software and virtual machines or to mimic network to external devices. Internal network uses virtual network devices that act as physical devices and enables single system to appear as a network.

   ii. External: used in Virtual local area networks (VLAN) and Virtual private networks (VPN).

(b) Application virtualization: this technique separates the application layer from the underlying operating system layer which means that application runs in 'encapsulated' state independently from operating system. Application is put into a capsule into which is then put copies of al shared resources that application would need to run as well as all DLLs (dynamicaly linked libraries), driver, registry entries and so on. In practical terms it means that application created for certain operating system can run on a different one, for example native Windows applications can be run under Linux distribution or vice-versa, we can isolate suspicious or malicious software and inspect it without the danger of infectiong whole system, run simultaneously applications that would otherwise conflict each other, easily deploy applications and so on. Some well known examples are Wine which is used to run Microsoft Windows applications on Linux, ThinApp from VMware, Xenocode from Code System Corporations and others.

(c) Desktop virtualization: similarly as application virtualization it separates desktop environment from underlying physical computer. Desktop virtualization functions on a client-server model. Clinet desktop environments are running on virtual machines stored on servers and client can access them via client device which can be standard PC or thin client. This technique is growing on popularity mainly because of cloud computing. It allows us to access desktop from any device or location (in practical terms it means that we can work in same environment from anywhere without the need to bring 'work computer'). By isong desktop virtalization we can use cheaper client desktop devices because far less processing power is required. Of course, businesses need to first invest into server hardware so it is capable of storing and streaming desired quantities of desktop environments but will save money in a long run by cheaper user devices. Desktop environments are stored on central server so they can be also centrally managed and controlled which increases security of whole system. Disadvantage is that streaming desktop environments to plenty of end users is very demanding on network infrastracture. Desktop virtualization is mostly used by companies with a lot of off-shore employees.

(d) Storage virtualization: multiple separated hardware storage devices (that can be on different physical locations) abstracted into single pool of virtualized storage space that act as single sotrage device localy connected to the computer. Storage virtualization is used to ignore differences between individual storage devices and simplify using them. It is ofthen used for back-ups and archives and can be implemented with software or hybrid software-hardware solutions. Common examples are:

i. NAS: (Network-attached storage) is server dedicated to managing storage space. NAS devices have to be part of LAN and they can be added or removed dynamically meaning that after adding/removing device whole system doesn't need to be restarted but it continues functioning as if nothing have happened.

ii. SAN: (Sotrage area network) is basically a sub-network containing only storage devices. Storage space managet by SAN can be accessed by any server in LAN or WAN. When new storage devices is added it is immediately available to any server in network. In practice SAN enables anyone within network have access to network's whole storage space.

Kapitola 2

Virtualization software

In this chapter we will go through well known virtualization implementations divided to virtualization architectures.

1. Xen Project
2. KVM
3. VirtualBox
4. UML
5. Docker Container
6. Wine

2.1 Xen Project

Is well known type I hypervisor which means it runs directly on the host hardware. Xen Project is widely used as base for a number of open-source and commercial applications providing server virtualization, desktop virtualization, infrastracture as a service (IaaS), security aplications, embedded and hardware appliances and so on. Worlds bigest clouds today also run on Xen Project hypervisor base. All operating systems based on recent Linux kernel are capable of running Xen project and have packages containing hypervisor and basic tools.

Xen is managed by a special priviledged virtula machine called Domain-0 or Dom0, priviledged means that it has device drivers and direct access to physical hardware. Domain-0 is a specially modified Linux kernel which is started by Xen hypervisor during initial stage of system start-up(OPISANE). Its role is to manage and control every other unpriviledged virtual machines (also called Domain-Us or DomU) that are running on the hypervisor. Domain-0 exposes control interface to the user and Xen project hypervisor cannot run without it. Through user interface in form of toolstack (or control stack) user can create, destroy or configure virtual machines, toolstack can by driven by command line console, graphical interface or cloud orchestration stack (for example OpenStack or CloudStack). [OBRAZOK]

Originally Xen only supported Paravirtualization (see link Paravirtualization). Support for Domain-U running in paravirtualized state is now included within upstream Linux kernel but support fo Domain-0 is not which means that it is easier to use Linux machine as a guest than as a host. Nowadays Xen supportste new virtualization processor extension added to the x86 architecture(aj tu), this is known as Xen as Hardware Virtual Machine (HVM).

HVM allows unmodified guest operating system to be virtualized on Xen hypervisor(aj tu) but it requires a special processors that support hardware virtualizaton extensions (Intel VT, AMD-V). These extencions allow for many of the priviledged kernel instructions to be handled directly by hardware using 'trap-and-emulate' techique, these were previously in paravirtualization converted to hypercalls.

Trap-and-emulate: Operating systems running on top of the hypervisor are run as user-level processes. They are not running at the same level of privilege as a Linux operating system that is running on bare metal. But if the operating system code is unchanged, it doesn't know that it does not have the privilege for doing certain things that it would do normally on bare metal hardware. In other words, when the operating system executes some privileged instructions, meaning they have to be in a privileged mode or kernel mode to run on bare metal in order to execute those instructions, those instructions will create a trap that goes into the hypervisor and the hypervisor will then emulate the intended functionality of the operating system. This is what is called the trap and emulate strategy. That is in some architectures, some privilege instructions may fail silently which means that you would think that the instruction actually succeeded, but it did not, and you may never know about it.

Here are key features of Xen project hypervisor:

1. Minimal footprint: around 1 MB. Xen uses microkernel design which minimalizes memory footprint and interface to the guest and is also more robust and secure than other types of hypervizors.
2. Driver isolation: hypervisor allows for the main device driver to run inside of a virtual machine. This is useful because if driver crashes it does not affect any other part of a system and virtual machine in which it runs can be rebooted and the driver restarted.
3. Many operating systems can be used: although most installaions use Linux as the domain-0 many other operating systems can by used such as NetBSD, OpenSolaris and others.

2.2 KVM

[obrazok] KVM - Kernel-based Virtual Machine lesser known virtualization solution then Xen Project. It has host and guest support in an upstream Linux kernel released in early 2007. KVM is kernel module which when loaded turns host kernel into type I hypervisor. To run it requires Intel VT or AMD-V extensions and enabled on a host system. By converting host machine kernel into hypervisor KVM can take advantage of already implemented components instead of implementing them from the scratch, for example it uses memory manager, scheduler, I/O stack, device drivers, security manager, network manager and others. In comparason to Xen architecture which requires maintenance of both Xen hypervisor and Domain-0, KVM is loadable kernel module and is easier to patch and upgrade. From host's perspective, every virtual machine is standard linux process and is treated as such.

Features:

1. Security: to improve security of virtual machines even further, KVM uses these approaches:
   (a) Security-enhanced Linux (SELinux) which establishes security boundaries around virtual machines.

   (b) Secure virtualization (sVirt) which boosts SELinux's capabilities and allow Mandatory Access Control (MAC) security to be applied.

2. Live migration: KVM supports live migration(see live migration) of virtual machines.
3. Scheduling and resource control: every virtual machine is seen as a standard process which means that Linux scheduler allow full control over resources allocated by it and guarantees quality of service. KVM offers completely fair scheduler, control groups, network name spaces and real-time extensions.
4. Storage: KVM supports shared file system which means that virtual machines can be shared by multiple hosts. Disk images support thin provisioning. Thin provisioning means that memory is allocated for virtual machine up to provisioned amount only when it needs it (Xen Project doesn't support thin provisioning, when vitual machine is provisioned for example 2 GB of RAM, after it starts, 2 GM of RAM are immediately allocated and can't be used elsewhere). This can lead to 'memory overcommit', state where more memory is assigned to virtual machines than is available on the system. KVM deals with memory overcommit in various ways:
   (a) Host can choose memory pages and write them to the disk. This leads to redcing performance as when virtual machine wants to access memory, host needs to read it from the disk which is significantly slower than RAM.
   (b) With VirtIO drivers, hot can request virtual machines to shrink their cache memory in order to free as much space as needed. This is called 'ballooning' and requires cooperation among host and guests.
   (c) KSM (Kernel Samepage Merging) is a process of merging identical memory pages from multiple virtual machines into a snigle read-only memory chunk while removing all duplicates of it. If any guest needs to write into one of merged pages, host creates writable copy which guest can modify.
5. Lower latency: kernel divides processes with long computing times into smaller pieces which are scheduled and processes acordingly.

2.3 VirtualBox

VirtualBox is a type II hypervisor currently being developed by Oracle Corporation. Oracle VM VirtualBox runs on Microsoft Windows, Mac OS X, Linux and Oracle Solaris systems and supports wide range of guest operating systems. With thousands of downloads each day it is the most popular cross-platform open-source virtualization solution.

Here are some of VirtualBox's main features:

1. Portability: VirtualBox has to run on an host operating system but its functionallity is to a very large degree identical on all of them, same files and image formats are used. This allow us to create a virtual machine on one host and run it on another host with different operating system. Virtual machines can be imported and exported using an Open Virtualization Format (OVP) with which we can import virtual machines created with different virtualization software.
2. No hardware virtualization needed: VirtualBox doesn't require a processor support like Intel VT or AMD-V so it can be run even on older hardware not possessing those features.
3. Guest additions: guest additions are software packages that can be installed inside of virtual machines to improve their performance or improve their integration with

host system. They consist of device drivers and system applications, for example on of guest additions is 'Shared folders addition' which provides an easy way to exchange files between host and guest. We can create a folder on host system and share it to the guest.

4. Hardware support:
   (a) Guest multiprocessing: VirtualBox can present up to 32 virtual CPUs to every virtual machine regardless of how many CPUs are present in host system.
   (b) USB device support: virtaul USB controller allows to connect USB device to virtual machine without a need to install device-specific drivers to the host machine.
   (c) ACPI support: Advanced Configuration and Power Interface is an open standard that operating systems can use to configure hardware components and to perform power management and status monitoring.
   (d) Built-in iSCSI support: this allows us to connect from virtual machine directly to the iSCSI storage server without going through host system which highly reduces overhead.
   (e) PXE support: Preboot eXecution Environment (PXE) in short is a way to boot operating system from a server on a virtual machine. Advantages are obvious, we don't need to have a operating system on a hard drive connected to the virtual machine, we just need to connect to server and boot it from there.
5. Snapshots: we can create snapshots of the current state of a virtual machine and store it. When needed we can reverse current state of VM and load configuration from any snapchot. This way we can periodicly save backups for quick recovery in case of emergency.
6. Grouping: multiple virtual machines can be collected into a group. We can than perform same operations over the group as we can over individual virtual machines (for example start, pause, shutdown, close, ...). By using groups we can manage multiple virtual machines with same configuration, purpose, etc at the same time as well as we can still manage individual VMs that are part of a group. One virtual machine can be inside multiple groups and groups can be nested into hierarchy.
7. Remote machine display: VRDE - VirtualBox Remote Desktop Extension allows for a high-performance remote access to any running virtual machine.

2.4 UML

[obrazok] User Mode Linux (UML) allows us to run Linux kernels as user mode processes under a host Linux kernel thus allowing us to run multiple independent virtual machines. Main difference between UML and other virtualization technologies is that UML is more of a virtual OS than virtual machine. Other solutiuons like VMWare are real virtual machines in that they emulate physical hardware and any operating system that runs on physical platform can also run on emulated one. Advantage of this solution is that guest OS is host OS-independent, meaning that any OS able to run on hardware is able to run on top of VMWare. On the other hand, UML is basicaly just Linux kernel modified to run in user space, UML guest can run only on Linux platform which is serious limitation but being more of virtual OS has other advantages. Solutions such as Xen, BSD jail or Solaris zones are integrated into host operating system but UML runs as a process. This has some performance costs but gives UML host OS version independence. UML has many real-world

uses but it’s most popular use-case is kernel development and debugging as it was its original purpose, for that can be used normal process-level tools like gdb, gporf (profiling) or gcov (coverage testing). Another popular uses are driver development, safe kernel testing and education due its simpler nature than other solutions.

2.5 Docker Container

Docker container is a operating system level technology established and promoted by Docker Inc. Docker container is operated by command-line tool called the Docker client which can run on the container host or through a remote interface connected to the container host. The main task of a Docker client is to pull images of containers from registry. Registry can be public or private and it is a repository of sources for ’ready to run’ virtual workloads. Main public registry is Docker Hub which is operated by Docker Inc. but nowadays there are planty of others. We can pull a container image using Docker daemon and from that image we can build working model for that container. A container is launched by running an image. An image is an executable package that includes everything needed to run an application–the code, a runtime, libraries, environment variables, and configuration files. Images that are mostly the same, except for the last few steps, can reduce disk usage by sharing parent layers. A container is a runtime instance of an image–what the image becomes in memory when executed (that is, an image with state, or a user process). Image can also include directives for daemon to preload the container with other components prior to running or directives for the loca command line after the local container image is build. The model of images and registries created standardized ways to build, load and manage containerized applications. Docker has been very successful in building a large open-source community which has contributed to the rising number of images in public and private repositories which attracts even more developers and enlarges open-source community. Docker image is defined by text-based Dockerfile which specifies a vase operating system image to start from, commands to prepare/build the image and commands to call whne image is ’run’. (Docker runs multiple containerized workloads on the same OS. By using containers, only the programs and their immediate dependencies are hosted by containers, with critical resources provided by the underlying operating system. This means that containerized systems can load applications faster and consume less resources.). Docker is available on many different operatin systems including most modern Linux distributions, Mac OSX and Windows.

OCI: The Open Container Initiative (OCI) is a lightweight, open governance structure (project), formed under the auspices of the Linux Foundation, for the express purpose of creating open industry standards around container formats and runtime. The OCI was launched on June 22nd 2015 by Docker, CoreOS and other leaders in the container industry [citovane]. The OCI currently contains two specifications: the Runtime Specification (runtime-spec) and the Image Specification (image-spec). The Runtime Specification outlines how to run a “filesystem bundle” that is unpacked on disk. [tiez]

Docker Engine is a client-server application with these major components:

1. A server which is a type of long-running program called a daemon process
2. A REST API which specifies interfaces that programs can use to talk to the daemon
3. A command line interface (CLI) client

Docker uses a client-server architecture. The Docker client talks to the Docker daemon, which does the heavy lifting of building, running, and distributing Docker containers. The

Docker client and daemon can run on the same system, or you can connect a Docker client to a remote Docker daemon. The Docker client and daemon communicate using a REST API. A Docker registry stores Docker images. Docker Hub and Docker Cloud are public registries and Docker is configured to look for images on Docker Hub by default. We can run our own private register. We can also buy or sell Docker images or distribute them for free in Docker store.

Docker uses a technology called namespaces. When we run a container, Docker creates a set of namespaces for it. These namespaces provide a layer of isolation. Each aspect of a container runs in a separate namespace and its access is limited to that namespace. On Linux, Docker Engine uses these namespaces:

1. pid: process isolation
2. net: managing network interfaces
3. ipc: managing access to IPC resources
4. mnt: managing filesystem mount points
5. uts: isolationg kernel and version identifiers

2.6 Wine

Wine - Wine Is Not an Emulator acronym means that Wine is not a virtual machine, it doeas not emulate physical hardware and we are not supposed to install Windows or any Windows driver on top of it. Different software programs are designed for different operating systems and are generaly not compatible with other operating systems, for example Windows programs can't run on Linux system because they use instructions that Linux system doesn't understand thus cannot interpret them, this is the main motivation for Wine. Wine is an implementation of Windows API and can be used as a library to port Windows applications to Unix, basicaly acting as a bridge between the two. It is a compatibility layer, everytime a Windows program tries to perform an action that Linux doesn't recognize, Wine will transalte it into one that it does. Wine can also recompile Windows program source code into format understandable for Linux. Even in recompiled form, Wine is still needed to run the program but there are many performance and other advantages to this process. Wine is an open source project.

Features: Wine is constantly growing in the features it supports, here are some of them:

1. Support for running, Win64, Win32, Win16 and Dos programs.
2. Optional use of external vendor DLLs.
3. MacOS and Android graphics support.
4. DirectX for games.
5. Support for alternative input devices such as graphics tablets
6. Winsock TCP/IP networking support.
7. Advances Unicode and foreign language support.
8. Fill featured Wine debugger and configuratable trace logging messages for easier troubleshooting.

Executables: Wines main task is to run Windows executebles, here are supported types:

1. DOS executable: very old programs for MS-DOS.

2. Windows NE executable: (NE - New Executable) They were the native processes run by Windows 2.x and 3.x.
3. Windows PE executable: (PE - Portable Executable) Introduced by Windows 95 and became the standard format for all later Windows versions. Portable Executable means that format of the esxecutable is independent of the CPU, even if the code IS dependent of the CPU.
4. Winelib executable: aplications written by using the Windows API but compiled as a Unix executable.

Wine architecture is close to the Windows NT architecture but several subsystems are not implemented yet. [obrazky]

Kapitola 3

Nikdy to nebude naprosto dokonalé

asdf

Kapitola 4

Závěr

Závěrečná kapitola obsahuje zhodnocení dosažených výsledků se zvlášť vyznačeným vlastním přínosem studenta. Povinně se zde objeví i zhodnocení z pohledu dalšího vývoje projektu, student uvede náměty vycházející ze zkušeností s řešeným projektem a uvede rovněž návaznosti na právě dokončené projekty.

Příloha A

Jak pracovat s touto šablonou

V této kapitole je uveden popis jednotlivých částí šablony, po kterém následuje stručný návod, jak s touto šablonou pracovat.

Jedná se o přechodnou verzi šablony. Nová verze bude zveřejněna do konce roku 2017 a bude navíc obsahovat nové pokyny ke správnému využití šablony, závazné pokyny k vypracování bakalářských a diplomových prací (rekapitulace pokynů, které jsou dostupné na webu) a nezávazná doporučení od vybraných vedoucích, která již teď najdete na webu (viz odkazy v souboru s literaturou). Jediné soubory, které se v nové verzi změní, budou `projekt-01-kapitoly-chapters.tex` a `projekt-30-prilohy-appendices.tex`, jejichž obsah každý student vymaže a nahradí vlastním. Šablonu lze tedy bez problémů využít i v současné verzi.

Popis částí šablony

Po rozbalení šablony naleznete následující soubory a adresáře:

bib-styles Styly literatury (viz níže).

obrazky-figures Adresář pro Vaše obrázky. Nyní obsahuje placeholder.pdf (tzv. TODO obrázek, který lze použít jako pomůcku při tvorbě technické zprávy), který se s prací neodevzdává. Název adresáře je vhodné zkrátit, aby byl jen ve zvoleném jazyce.

template-fig Obrázky šablony (znak VUT).

fitthesis.cls Šablona (definice vzhledu).

Makefile Makefile pro překlad, počítání normostran, sbalení apod. (viz níže).

projekt-01-kapitoly-chapters.tex Soubor pro Váš text (obsah nahraďte).

projekt-20-literatura-bibliography.bib Seznam literatury (viz níže).

projekt-30-prilohy-appendices.tex Soubor pro přílohy (obsah nahraďte).

projekt.tex Hlavní soubor práce – definice formálních částí.

Výchozí styl literatury (czechiso) je od Ing. Martínka, přičemž anglická verze (englishiso) je jeho překladem s drobnými modifikacemi. Oproti normě jsou v něm určité odlišnosti, ale na FIT je dlouhodobě akceptován. Alternativně můžete využít styl od Ing. Radima Loskota nebo od Ing. Radka Pyšného[1]. Alternativní styly obsahují určitá vylepšení, ale zatím nebyly

[1] BP Ing. Radka Pyšného `http://www.fit.vutbr.cz/study/DP/BP.php?id=7848`

řádně otestovány větším množstvím uživatelů. Lze je považovat za beta verze pro zájemce, kteří svoji práci chtějí mít dokonalou do detailů a neváhají si nastudovat detaily správného formátování citací, aby si mohli ověřit, že je vysázený výsledek v pořádku.

Makefile kromě překladu do PDF nabízí i další funkce:

- přejmenování souborů (viz níže),
- počítání normostran,
- spuštění vlny pro doplnění nezlomitelných mezer,
- sbalení výsledku pro odeslání vedoucímu ke kontrole (zkontrolujte, zda sbalí všechny Vámi přidané soubory, a případně doplňte).

Nezapomeňte, že vlna neřeší všechny nezlomitelné mezery. Vždy je třeba manuální kontrola, zda na konci řádku nezůstalo něco nevhodného – viz Internetová jazyková příručka[2].

Pozor na číslování stránek! Pokud má obsah 2 strany a na 2. jsou jen „Přílohy“ a „Seznam příloh“ (ale žádná příloha tam není), z nějakého důvodu se posune číslování stránek o 1 (obsah „nesedí“). Stejný efekt má, když je na 2. či 3. stránce obsahu jen „Literatura“ a je možné, že tohoto problému lze dosáhnout i jinak. Řešení je několik (od úpravy obsahu, přes nastavení počítadla až po sofistikovanější metody). **Před odevzdáním proto vždy překontrolujte číslování stran!**

Doporučený postup práce se šablonou

1. **Zkontrolujte, zda máte aktuální verzi šablony.** Máte-li šablonu z předchozího roku, na stránkách fakulty již může být novější verze šablony s aktualizovanými informacemi, opravenými chybami apod.
2. **Zvolte si jazyk**, ve kterém budete psát svoji technickou zprávu (česky, slovensky nebo anglicky) a svoji volbu konzultujte s vedoucím práce (nebyla-li dohodnuta předem). Pokud Vámi zvoleným jazykem technické zprávy není čeština, nastavte příslušný parametr šablony v souboru projekt.tex (např.: `documentclass[english]{fitthesis}` a přeložte prohlášení a poděkování do angličtiny či slovenštiny.
3. **Přejmenujte soubory.** Po rozbalení je v šabloně soubor `projekt.tex`. Pokud jej přeložíte, vznikne PDF s technickou zprávou pojmenované `projekt.pdf`. Když vedoucímu více studentů pošle `projekt.pdf` ke kontrole, musí je pracně přejmenovávat. Proto je vždy vhodné tento soubor přejmenovat tak, aby obsahoval Váš login a (případně zkrácené) téma práce. Vyhněte se však použití mezer, diakritiky a speciálních znaků. Vhodný název může být např.: „`xlogin00-Cisteni-a-extrakce-textu.tex`“. K přejmenování můžete využít i přiložený Makefile:

   ```
   make rename NAME=xlogin00-Cisteni-a-extrakce-textu
   ```
4. Vyplňte požadované položky v souboru, který byl původně pojmenován `projekt.tex`, tedy typ, rok (odevzdání), název práce, svoje jméno, ústav (dle zadání), tituly a jméno vedoucího, abstrakt, klíčová slova a další formální náležitosti.
5. Nahraďte obsah souborů s kapitolami práce, literaturou a přílohami obsahem svojí technické zprávy. Jednotlivé přílohy či kapitoly práce může být výhodné uložit do samostatných souborů – rozhodnete-li se pro toto řešení, je doporučeno zachovat konvenci pro názvy souborů, přičemž za číslem bude následovat název kapitoly.

[2] Internetová jazyková příručka `http://prirucka.ujc.cas.cz/?id=880`

6. Nepotřebujete-li přílohy, zakomentujte příslušnou část v `projekt.tex` a příslušný soubor vyprázdněte či smažte. Nesnažte se prosím vymyslet nějakou neúčelnou přílohu jen proto, aby daný soubor bylo čím naplnit. Vhodnou přílohou může být obsah přiloženého paměťového média.
7. Nascanované zadání uložte do souboru `zadani.pdf` a povolte jeho vložení do práce parametrem šablony v projekt.tex (`documentclass[zadani]{fitthesis}`).
8. Nechcete-li odkazy tisknout barevně (tedy červený obsah – bez konzultace s vedoucím nedoporučuji), budete pro tisk vytvářet druhé PDF s tím, že nastavíte parametr šablony pro tisk: (`documentclass[zadani,print]{fitthesis}`). Barevné logo se nesmí tisknout černobíle!
9. Vzor desek, do kterých bude práce vyvázána, si vygenerujte v informačním systému fakulty u zadání. Pro disertační práci lze zapnout parametrem v šabloně (více naleznete v souboru fitthesis.cls).
10. Nezapomeňte, že zdrojové soubory i (obě verze) PDF musíte odevzdat na CD či jiném médiu přiloženém k technické zprávě.

Obsah práce se generuje standardním příkazem `\tableofcontents` (zahrnut v šabloně). Přílohy jsou v něm uvedeny úmyslně.

Pokyny pro oboustranný tisk

- **Oboustranný tisk je doporučeno konzultovat s vedoucím práce.**
- Je-li práce tištěna oboustranně a její tloušťka je menší než tloušťka desek, nevypadá to dobře.
- Zapíná se parametrem šablony: `\documentclass[twoside]{fitthesis}`
- Po vytištění oboustranného listu zkontrolujte, zda je při prosvícení sazební obrazec na obou stranách na stejné pozici. Méně kvalitní tiskárny s duplexní jednotkou mají často posun o 1–3 mm. Toto může být u některých tiskáren řešitelné tak, že vytisknete nejprve liché stránky, pak je dáte do stejného zásobníku a vytisknete sudé.
- Za titulním listem, obsahem, literaturou, úvodním listem příloh, seznamem příloh a případnými dalšími seznamy je třeba nechat volnou stránku, aby následující část začínala na liché stránce (\cleardoublepage).
- Konečný výsledek je nutné pečlivě překontrolovat.

Styl odstavců

Odstavce se zarovnávají do bloku a pro jejich formátování existuje více metod. U papírové literatury je častá metoda s použitím odstavcové zarážky, kdy se u jednotlivých odstavců textu odsazuje první řádek odstavce asi o jeden až dva čtverčíky (vždy o stejnou, předem zvolenou hodnotu), tedy přibližně o dvě šířky velkého písmene M základního textu. Poslední řádek předchozího odstavce a první řádek následujícího odstavce se v takovém případě neoddělují svislou mezerou. Proklad mezi těmito řádky je stejný jako proklad mezi řádky uvnitř odstavce. [**?**] Další metodou je odsazení odstavců, které je časté u elektronické sazby textů. První řádek odstavce se při této metodě neodsazuje a mezi odstavce se vkládá vertikální mezera o velikosti 1/2 řádku. Obě metody lze v kvalifikační práci použít, nicméně často je vhodnější druhá z uvedených metod. Metody není vhodné kombinovat.

Jeden z výše uvedených způsobů je v šabloně nastaven jako výchozí, druhý můžete zvolit parametrem šablony „`odsaz`“.

Užitečné nástroje

Následující seznam není výčtem všech využitelných nástrojů. Máte-li vyzkoušený osvědčený nástroj, neváhejte jej využít. Pokud však nevíte, který nástroj si zvolit, můžete zvážit některý z následujících:

MikTeX LaTeX pro Windows – distribuce s jednoduchou instalací a vynikající automatizací stahování balíčků.

TeXstudio Přenositelné opensource GUI pro LaTeX. Ctrl+klik umožňuje přepínat mezi zdrojovým textem a PDF. Má integrovanou kontrolu pravopisu, zvýraznění syntaxe apod. Pro jeho využití je nejprve potřeba nainstalovat MikTeX.

WinEdt Ve Windows je dobrá kombinace WinEdt + MiKTeX. WinEdt je GUI pro Windows, pro jehož využití je nejprve potřeba nainstalovat MikTeX či TeX Live.

Kile Editor pro desktopové prostředí KDE (Linux). Umožňuje živé zobrazení náhledu. Pro jeho využití je potřeba mít nainstalovaný TeX Live a Okular.

JabRef Pěkný a jednoduchý program v Javě pro správu souborů s bibliografií (literaturou). Není potřeba se nic učit – poskytuje jednoduché okno a formulář pro editaci položek.

InkScape Přenositelný opensource editor vektorové grafiky (SVG i PDF). Vynikající nástroj pro tvorbu obrázků do odborného textu. Jeho ovládnutí je obtížnější, ale výsledky stojí za to.

GIT Vynikající pro týmovou spolupráci na projektech, ale může výrazně pomoci i jednomu autorovi. Umožňuje jednoduché verzování, zálohování a přenášení mezi více počítači.

Overleaf Online nástroj pro LaTeX. Přímo zobrazuje náhled a umožňuje jednoduchou spolupráci (vedoucí může průběžně sledovat psaní práce), vyhledávání ve zdrojovém textu kliknutím do PDF, kontrolu pravopisu apod. Zdarma jej však lze využít pouze s určitými omezeními (někomu stačí na disertaci, jiný na ně může narazit i při psaní bakalářské práce) a pro dlouhé texty je pomalejší.

Pozn.: Overleaf nepoužívá Makefile v šabloně – aby překlad fungoval, je nutné kliknout pravým tlačítkem na `projekt.tex` a zvolit „Set as Main File“.

Užitečné balíčky pro LaTeX

Studenti při sazbě textu často řeší stejné problémy. Některé z nich lze vyřešit následujícími balíčky pro LaTeX:

- `amsmath` – rozšířené možnosti sazby rovnic,
- `float, afterpage, placeins` – úprava umístění obrázků,
- `fancyvrb, alltt` – úpravy vlastností prostředí Verbatim,
- `makecell` – rozšíření možností tabulek,
- `pdflscape, rotating` – natočení stránky o 90 stupňů (pro obrázek či tabulku),
- `hyphenat` – úpravy dělení slov,
- `picture, epic, eepic` – přímé kreslení obrázků.

Některé balíčky jsou využity přímo v šabloně (v dolní části souboru fitthesis.cls). Nahlédnutí do jejich dokumentace může být rovněž užitečné.

Sloupec tabulky zarovnaný vlevo s pevnou šířkou je v šabloně definovaný „L“ (používá se jako „p“).