Operētajsistēmu inženierija

Kursu vada Leo Seļāvo

Kursa apraksts

- Skat. tiešsaistē
  - Mājas (lasāmā) viela
  - ~22 lekcijas, 2 kontroldarbi
  - Praktiskie darbi klasē un “mājās”

- Operētājsistēmu uzbūve un elementi
  - virtual memory,
  - kernel and user mode,
  - system calls,
  - threads,
  - context switches,
  - interrupts,
  - interprocess communication,
  - coordination of concurrent activities,
  - the interface between software and hardware

Praktiskie darbi

- Praktiskie darbi
  - Būvējam primitīvu OS, pēc UNIX xv6 parauga
  - Papildinām ar dažiem jaunākajiem sasniegumiem

- Sastāvdaļas
  - Booting
  - Memory Management
  - User-level Environments
  - Preemptive Multitasking
  - File System and Spawn
  - A Shell

Atzīme

- 20%
  - Dalība klasē un mājas darbi
- 30%
  - Divi kontroldarbi
- 50%
  - Projekts un laboratorijas darbi

1. lekcija: OS pārskats

- Goal of course:
  - Understand operating systems in detail by designing and implementing miminal OS
  - Hands-on experience with building systems

What is an operating system?

- a piece of software that turns the hardware into something useful
- layered picture: hardware, OS, applications
- Three main functions: fault isolate applications, abstract hardware, manage hardware

Piemēri

- OS-X, Windows, Linux, *BSD, ... (desktop, server)
- PalmOS, Windows/CE/Mobile (PDA)
- Symbian, JavaOS (Cell phones)
- VxWorks, pSOS (real-time)
- TinyOS, Contiki, LiteOS, MansOS (wireless sensor networks)
- …

OS abstrakcijas

- processes: fork, wait, exec, exit, kill, getpid, brk, nice, sleep, trace
- files: open, close, read, write, lseek, stat, sync
- directories: mkdir, rmdir, link, unlink, mount, umount
- users + security: chown, chmod, getuid, setuid
- interprocess communication: signals, pipe
- networking: socket, accept, snd, recv, connect
- time: gettimeofday
- terminal:

Piemēri

- int read(int fd, void*, int)
- int write(int fd, void*, int)
- off_t lseek(int fd, off_t, int [012])
- int close(int fd)
- int fsync(int fd)
- int open(const char*, int flags [, int mode])
  - O_RDONLY, O_WRONLY, O_RDWR, O_CREAT
- mode_t umask(mode_t cmask)

Piemēri 2

- int mkdir(char *path, mode_t mode);
- DIR *opendir(char *dirname)
- struct dirent *readdir(DIR *dirp)
- int closedir(DIR *dirp)
- int chdir(char *path)
- int link(char *existing, char *new)
- int unlink(char *path)
- int rename(const char*, const char*)
- int rmdir(char *path)

Piemēri 3

- int stat(char *path, struct stat *buf)
- int mknod(char *path, mode_t mode, dev_t dev)
- int fork()
  - returns childPID in parent, 0 in child; only difference
- int getpid()
- int waitpid(int pid, int* stat, int opt)
  - pid==-1: any; opt==0||WNOHANG
  - returns pid or error
- void _exit(int status)
- int kill(int pid, int signal)
- int sigaction(int sig, struct sigaction *, struct sigaction *)
- int sleep (int sec)
- int execve(char* prog, char** argv, char** envp)
- void *sbrk(int incr)

Piemēri 4

- int dup2(int oldfd, int newfd)
- int fcntl(int fd, F_SETFD, int val)
- int pipe(int fds[2])
  - writes on fds[1] will be read on fds[0]
  - when last fds[1] closed, read fds[0] retursn EOF
  - when last fds[0] closed, write fds[1] kills SIGPIPE/fails EPIPE
- int fchown(int fd, uind_t owner, gid_t group)
- int fchmod(int fd, mode_t mode)
- int socket(int domain, int type, int protocol)
- int accept(int socket_fd, struct sockaddr*, int* namelen)
  - returns new fd
- int listen(int fd, int backlog)
- int connect(int fd, const struct sockaddr*, int namelen)
- void* mmap(void* addr, size_t len, int prot, int flags, int fd, off_t offset)
- int munmap(void* addr, size_t len)
- int gettimeofday(struct timeval*)

Kursa struktūra

- Lab: minimal OS for x86 in an exokernel style (50%)
  - kernel interface: hardware + protection
  - libOS implements fork, exec, pipe, ...
  - applications: file system, shell, ..
  - development environment: gcc, bochs
  - lab 1 is out (**online**)

Kursa struktūra 2

- Lekcijas (20%)
  - Mājas darbs
  - 1. puse – lekcija
  - 2. puse – “case study”
- Kontroldarbi
  - Semestra vidū
  - Semestra beigās (eksāmens)
  - Izstrādatās OS prezentācija

Case Study

- The Shell (skat lekcijas piezīmes)