OS 2018

MIT ;)

https://pdos.csail.mit.edu/6.828/2018

Struktura prednasky

- 1. uvod
- 2. predmet
- 3. uvod do os
- 4. start pc

1. cast: UVOD

Ciele predmetu

- Detailnejsie pochopenie OS
- Ako:
  - Navrh
  - Implementacia
- Vlastny (malilinkaty) OS ;)

Naco je dobry OS

- Aplikacie (izolacia ↔ zdielanie)
- Sluzby (hw → app, spravodlivost)
- Hardver (vykon)

Co chcu aplikacie od OS

- Pristup k hardveru, co vyzaduje:
  - Abstrakciu hardveru (problem ovladacov)
  - Multiplexovanie hw medzi aplikaciami
- Izolovanie poskodenych aplikacii od zvysku OS
- Komunikaciu medzi aplikaciami

Ake sluzby od OS ocakavame

- Procesy
- Pamat
- Pristup ku suborom
- Adresarova struktura
- Bezpecnost
- Siet
- Viaceri pouzivatelia
- Komunikacia medzi aplikaciami (IPC)
- ...

Co znamena abstrakcia hw?

- Pristup ku hw spravovany jadrom na zaklade ziadosti aplikacie
- Jednotne rozhranie k roznym zariadeniam toho isteho typu (zbernica)
- Programator ovladaca sa moze sustredit iba na jednu cast OS

Ako vyzera abstrakcia OS?

- Aplikacie vyuzivaju sluzby jadra cez tzv. systemove volania (syscalls)
- Priklad z Unix systemov:

```
fd = open(“subor.txt”,1);
write(fd, “ahoj svjete!\n”, 13);
pid = fork();
```

Preco sa venovat takejto oblasti?

- Sklbenie vedomosti: AlPr, PT, OOP
- Cielom je napisat program (jadro), ktory:
  - Musi byt extremne efektivny (rychlost), ale na druhej strane dostatocne abstraktny (prenositelny na ine platformy)
  - Musi byt vykonny (vela sluzieb a moznosti), ale pritom jednoduchy (zlozeny z jednoducho nahraditelnych blokov)

Preco sa venovat takejto oblasti?

- Naucit sa mysliet, spravne navrhovat a implementovat algoritmy, ktore medzi sebou spolupracuju
- Vyuzit vsetky doteraz ziskane vedomosti a sklbit ich
- OS je najkomplexnejsi program vobec

Linux 4.8-rc7 19.9.2016

```
y@ellYah:/usr/src/linux-4.8-rc7$ cloc .
   55460 text files.
   54971 unique files.
    9923 files ignored.

http://cloc.sourceforge.net v 1.56  T=192.0 s (235.7 files/s, 106196.6 lines/s)
```

Language	files	blank	comment	code
C	23425	2258325	2074816	11503172
C/C++ Header	17771	433230	741250	2835111
Assembly	1433	48266	63102	294019
make	2217	8254	7915	34510
Perl	49	5183	3760	27188
Bourne Shell	201	2199	3513	11559
Python	44	1579	1917	9262
yacc	8	655	355	4327
HTML	3	512	0	4289
lex	8	299	289	1894
ASP.Net	19	133	0	1615
Bourne Again Shell	46	355	260	1585
C++	1	231	58	1581
awk	10	132	131	1138
NAnt scripts	2	128	0	475
Pascal	3	49	0	231
Lisp	1	63	0	218
Objective C++	1	55	0	189
m4	1	15	1	95
XSLT	6	13	27	71
vim script	1	3	12	27
CSS	1	12	22	25
sed	2	0	25	16
Teamcenter def	1	0	2	6
SUM:	45254	2759691	2897455	14732603

Linux 4.13.1 10.9.2017

```
y@ellYah:/usr/src/linux-4.13.1$ cloc .
   60545 text files.
   60002 unique files.
   11247 files ignored.

http://cloc.sourceforge.net v 1.56  T=211.0 s (232.3 files/s, 108178.4 lines/s)
```

Language	files	blank	comment	code
C	25241	2468666	2227885	12531579
C/C++ Header	19482	484063	907454	3641969
Assembly	1434	49263	64821	298076
make	2362	8582	8151	36990
Perl	50	5061	3707	26293
Bourne Shell	246	3091	4180	15853
Python	72	2099	2426	12018
HTML	3	565	0	4730
yacc	9	682	357	4530
lex	8	302	301	1906
C++	7	287	71	1838
ASP.Net	32	158	0	1808
Bourne Again Shell	47	384	312	1713
awk	12	185	170	1510
NAnt scripts	2	158	0	588
Pascal	3	49	0	231
Objective C++	1	55	0	189
m4	1	15	1	95
XSLT	5	13	26	61
CSS	1	14	23	35
vim script	1	3	12	27
Teamcenter def	1	0	2	6
sed	1	2	5	5
SUM:	49021	3023697	3219904	16582050

Linux 4.19-rc3 10.9.2018

```
y@ellYah:/mnt/data1/skola/os/2018/__prednasky/01/linux-4.19-rc3$ cloc .
   61684 text files.
   61262 unique files.
   12218 files ignored.

http://cloc.sourceforge.net v 1.56  T=225.0 s (219.2 files/s, 104909.5 lines/s)
```

Language	files	blank	comment	code
C	26078	2586063	2268480	13128894
C/C++ Header	18861	492135	903746	3659127
Assembly	1323	47331	61174	278057
make	2386	8738	9443	37957
Bourne Shell	376	6571	5894	28200
Perl	55	5426	4004	27407
Python	102	2775	2975	15979
HTML	5	670	0	5497
yacc	9	701	375	4648
lex	8	326	315	2006
ASP.Net	38	191	0	1981
C++	7	286	77	1844
Bourne Again Shell	51	352	318	1722
awk	11	170	155	1386
NAnt scripts	2	155	0	588
Teamcenter def	2	14	2	100
m4	1	15	1	95
XSLT	5	13	26	61
CSS	1	18	27	44
vim script	1	3	12	27
Ruby	1	4	0	25
sed	1	2	5	5
SUM:	49324	3151959	3257029	17195650

2016 → 2017

- +5000 novych suborov
- +220 000 riadkov komentarov
- +1 000 000 riadkov kodu v C
- +800 000 riadkov v hlavickovych suboroch C

- +4000 riadkov kodu v ASM !!!

2017 → 2018

- +1 500 novych suborov
- +37 125 riadkov komentarov
- +600 000 riadkov kodu v C
- +17 000 riadkov v hlavickovych suboroch C

- +20 000 riadkov kodu v ASM !!! (5x viac ako vlani)

Nami vyvijane jadro JOS

- Jednoduchucke, minimalisticke
- Po nastartovani funguje jednoduchy shell, pomocou ktoreho je mozne interagovat s jadrom
- Ciel: doplnat funkcionalitu
  - Sprava pamate (strankovanie)
  - Procesy
  - Subory
  - Siet
  - ...

JOS

```
y@ellYah:/mnt/data1/src/xv6/xv6-public.2016-09-06/lab$ cloc .
      47 text files.
      47 unique files.
      70 files ignored.

http://cloc.sourceforge.net v 1.56  T=0.5 s (68.0 files/s, 10430.0 lines/s)
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
C                               12            289            288           2292
C/C++ Header                    15            179            218            782
Python                           1             91             76            380
make                             1             54             46            218
Assembly                         2             37              1            143
Perl                             2             16             30             63
Lisp                             1              0              0             12
-------------------------------------------------------------------------------
SUM:                            34            666            659           3890
-------------------------------------------------------------------------------
```

2. cast: PREDMET

Struktura predmetu

- http://147.175.106.2/kaivt/Predmety/OS/
- Kurz podla MIT: https://pdos.csail.mit.edu/6.828/2018

- Praca na predmete:
  - Pocas cviceni vo dvojiciach
  - Hodnotenie kazdy samostatne

Struktura predmetu

- Prednasky:
  - Zakladna teoria OS
  - Pohlad na implementaciu systemu JOS
- Cvicenia:
  - JOS, maly OS pre x86
  - Doplnanie API: fork, exec, pipe...
  - Doplnanie funkcionality jadra, uzivateske programy...

Co su hlavne casti JOS

- Startovanie (bootovanie)
- Sprava pamate
- API
- Preemptivny multitasking
- Suborovy system a shell
- Sietovy ovladac
- Zaverecny projekt podla vlastneho vyberu

Cvicenia a praca doma

- Praca na doma (DU) pozostava z 2 casti:
  - Priprava na dalsiu prednasku podla pokynov
  - Dopracovanie uloh z cvika, ktore sa na cviku nestihnu
- Ohodnocovanie DU:
  - Nahodne (napr. pisomkami alebo osobne)
  - Nepracovanie na ulohach pocas cvika → NZ

Pracovne prostredie

- VirtualBox + Ubuntu 16.04
- Qemu
- JOS, xv6
- (vid pokyny na stranke cviceni)

- Virtualky v C117, vitane vlastne stroje
- C117 wifi eduroam, malo el. zasuviek!!!

Hodnotenie (1)

- Akykolvek identifikovany pokus o podvod:
  - Disciplinarna komisia FEI STU
  - FX hodnotenie z predmetu
- Nutna (nie postacujuca) podmienka ziskania hodnotenia lepsieho nez FX
  - Vyplnenie evaluacie
  - Kontrola vyplnenia evaluacie individualne na konci semestra

Hodnotenie (2)

- Bonusy (kladne ci zaporne) podla lubovole prednasajuceho
- 2x 30b pisomny test pocas semestra
  - 6. tyzden (pondelok 22. oktobra o 17:00)
  - 12. tyzden (pondelok 10. decembra o 17:00)
- 2x 20b programatorsky test na cviku pocas semestra
  - 6. tyzden (22. a 23. oktobra na cvikach)
  - 12. tyzen (10. a 11. decembra na cvikach)

Obsah hodnotenia

- Cvicenia:
  - Chapanie teorie, pripravenost na cvicenie
  - Pripravenost zdrojakov
- Zapocty:
  - Dopracovanie kodu priamo na cviceni
- Testy:
  - Vedomosti
- Prednasky:
  - Pripravenost na prednasku podla pokynov

3. cast: UVOD DO OS

Ocakavania od OS

- Podpora viacerych veci SUCASNE
- Zdielanie a prerozdelovanie zdrojov hw (CPU pre procesy, pamat, disk, tlaciaren, mys, monitor, sietova karta...)
- Izolacia procesov (aby nemohol jeden nicit druhy len tak, z cirej zloby)
- Komunikacia procesov je vsak tiez potrebna

Poziadavky na OS

1) Multiplex
2) Izolovanie
3) Interakcia

Preco multiplex?

- Co keby sme nemali OS, ale kazda aplikacia by pomocou nejakej kniznice priamo pristupovala k hw, ktory potrebuje?
- Vid niektore vnorene (embedded) aplikacie

Preco multiplex?

- Ak by bola aplikacia jedina, OK
- Ak je ich viac, musela by byt kazda “slusna”, museli by spolupracovat (kooperativny multitasking)
- Aplikacie su casto plne chyb... a nie su vzdy slusne

Preco izolacia?

- Aby sa zachovala “spravodlivost” pri pridelovani hw, oddelime aplikacie
- Pristup nie k hw, ale k abstrakcii hw cez sluzbu OS (priklad: pristup na disk cez sys volania open(), read(), write(), close())
- Vyhody:
  - Abstraktnejsi pristup aplikacie (nemusi pouzivat cisla sektorov, ale mena suborov)
  - Nepride ku chybe pri praci s diskom

Preco interakcia?

- Ak mame izolovany beh aplikacie, ako by medzi sebou mohli priamo komunikovat?
- Nepriamo cez sluzby OS:
  - Suborove popisovace
  - Mapovanie pamate
  - Siet

Uzivatelsky priestor

- Silna izolacia si vyzaduje presne definovane API (pristupove body ku sluzbam jadra)
- Aplikacia by nemala byt schopna pristupovat k internym datovym strukturam a instrukciam jadra
- CPU poskytuje hw podporu takejto ochrany

User / Kernel space

- Moderne CPU poskytuju minimalne 2 rezimy cinnosti:
  - Kernel mod (vsetky instrukcie CPU povolene), napriklad priamy pristup k zariadeniam
  - User mod (nie vsetky instrukcie CPU povolene); ak aplikacia v tomto mode skusi vykonat privilegovanu instrukciu, vyvola sa vynimka a OS ma moznost “prehovorit do duse” tejto apke. Zvacsa tym, ze ju ukonci ;)

User / Kernel space

- Aplikacia beziaca v rezime “user mod” bezi v tzv. Priestore uzivatela (“user space”)

- Program beziaci v rezime CPU “kernel mod” sa vykonava v tzv. Priestore jadra (“kernel space”)
- V pripade OS sa takemuto programu vravi “jadro” - angl. “kernel”

•Co vsetko ma bezat v jadre?

- Vsetky sys volania? → monoliticke jadro
  - Vsetko potrebne v jadre
  - Rozne casti jadra mozu priamo komunikovat (napr. spolocny buffer pre virtualnu pamat a suborovy system)
  - Problem je prave to, ze vsetko je v jadre a vsetko moze/chce komunikovat so vsetkym → komplexne rozhrania, vnasanie chyb
  - V pripade chyby hrozi pad celeho jadra a tym systemu, vyzaduje sa restart pocitaca

•Co vsetko ma bezat v jadre?

- Minimalna funkcionalita - mikrokernel
  - Sluzby OS bezia v priestore uzivatela, vtedy sa taketo aplikacie volaju servery
  - Na vyuzivanie sluzieb serverov je definovane rozhranie, tzv. posielanie sprav serverom
  - Funkcionalita mikrojadra – spustanie aplikacii, posielanie sprav, spristupnovanie hw

Priklady z praxe

- Linux
  - Monoliticke jadro
  - Niektore sluzby v user space (graficky subsystem)
- Xv6
  - Monoliticke jadro
  - Tak malo sluzieb, ze jadro je mensie ako niektore mikrojadra ;)

4. cast: START PC

Start PC

- Fyzicka pamat
- Realny mod procesora

Start PC

- BIOS
  - Inicializacia hw
  - Posunut vykonavanie dalej zavadzacu OS
- Zavadzac
  - Prvy sektor (boot sector) media
  - Nahrava sa do pamate na adresu 0x7c00

Start PC

- Zavadzac (boot loader)
  - Prepina CPU do moderneho modu cinnosti
  - Nahrava program jadra z disku do pamate
  - Predava riadenie jadru
- Xv6 boot loader pozostava z 2 suborov
  - bootasm.S
  - bootmain.c

- Do buducej prednasky PRECITAT Appendix B z pracovnej knizky:

  https://pdos.csail.mit.edu/6.828/2018/xv6/book-rev11.pdf

MIT ;)