Systemy Operacyjne

Ćwiczenia

Sprawy organizacyjne

- Kontakt:
  - Artur.Basiura@agh.edu.pl
  - Konsultacje po uzgodnieniu mailowym terminu
  - **Prośba o kontakt mailowy do jednej osoby z grupy** (zmiany w terminach zajęć)

- Zasady współpracy: https://geist.re/pub:teaching:gris
- Instrukcje: http://ai.ia.agh.edu.pl/wiki/pl:dydaktyka:so:2017:start

- Literatura:
  - M.J.Rochkind - Programowanie w systemie UNIX dla zaawansowanych (*Advanced UNIX Programming*) - obowiązuje **wyd. 3 PL = wyd. 2 EN**

Zajęcia

- Zasady współpracy: https://geist.re/pub:teaching:gris

- Obecności obowiązkowe - 1 nieobecność nieusprawiedliwiona
- Możliwość odrabiania zajęć
- Student może nie zostać dopuszczony do zaliczenia poprawkowego w przypadku opuszczenia więcej niż 20% zajęć (powyżej 3 zajęć).

- 3 kolokwia
- Kartkówka z tematyki poprzednich zajęć i tematyki przyszłych zajęć (2 pytania)
- Możliwość uzyskania plusów na zajęciach za aktywność (1plus = 1 pkt)

- Zaliczenie > 50% max pkt.

Oceny

- **Punktacja**
  - **Kartkówka**: 2 pyt x 1pkt = 2pkt (planowo 10 kartkówek)
    - Na kartkówkach obowiązuje materiał z poprzedniego laboratorium oraz bieżącego.
  - **1. kolokwium**: 5 lab x 4pkt = 20pkt
  - **2. kolokwium**: 4 lab x 4pkt = 16pkt
  - **3. kolokwium**: 3 lab x 4pkt = 12pkt
  - Powyższe punkty stanowią 100% maksymalnej łącznej liczby punktów (MAX – 68 pkt)

- **Ocena** (wg Skali Ocen Regulaminu Studiów AGH §13 p.1)
  - >= 91% - bardzo dobry (5.0);
  - >= 81% - plus dobry (4.5);
  - >= 71% - dobry (4.0);
  - >= 61% - plus dostateczny (3.5);
  - >= 50% - dostateczny (3.0);
  - < 50% - niedostateczny (2.0).

Zajęcia 1 - Wprowadzenie

- 1. Organizacja przedmiotu
- 2. Dokumentacja systemu UNIX
- 3. Wykorzystanie funkcji systemowych i sprawdzanie błędów
- 4 Kompilowanie programów: gcc i make

- *5 Daty i czasy w Unixie (o ile czas pozwoli)*
- *6 Standardy systemu (o ile czas pozwoli)*

Instrukcje i zajęcia

- Instrukcje: http://ai.ia.agh.edu.pl/wiki/pl:dydaktyka:so:2017:start
  - User: so
  - Pass: $02017

- Harmonogram zajęć

L.p.	Wtorki	Środy	Czwartki	Instrukcja do laboratorium
1.	28.02.2017	01.03.2017	02.03.2017	Wprowadzenie
2.	07.03.2017	08.03.2017	09.03.2017	Podstawowe operacje na plikach
3.	14.03.2017	15.03.2017	16.03.2017	Zaawansowane operacje na plikach
4.	21.03.2017	22.03.2017	23.03.2017	Procesy 1
5.	28.03.2017	29.03.2017	30.03.2017	Procesy 2
6.	04.04.2017	05.04.2017	06.04.2017	1. kolokwium
7.	11.04.2017	12.04.2017	**20.04.2017**	Komunikacja międzyprocesowa 1
8.	25.04.2017	26.04.2017	27.04.2017	Komunikacja międzyprocesowa 2
9.	09.05.2017	10.05.2017	**04.05.2017**	Komunikacja międzyprocesowa 3
10.	16.05.2017	17.05.2017	**11.05.2017**	Wyjście terminalowe
11.	23.05.2017	24.05.2017	**18.05.2017**	2. kolokwium
12.	30.05.2017	31.05.2017	**25.05.2017**	Programowanie sieciowe 1
13.	06.06.2017	07.06.2017	**01.06.2017**	Programowanie sieciowe 2
14.	13.06.2017	14.06.2017	**08.06.2017**	Programowanie sieciowe 3 + 3. kolokwium[1)]

Dokumentacja systemu UNIX

- Dokumentacja: man, apropos, whatis, info
- man
  - Kategorie dokumentacji (8+ kategorii) - passwd(1) , passwd(5)
  - Sekcje stron w man (np. NAMES, SYNOPSIS, DESCRIPTION, ERROR,whati OPTIONS …)
  - Konwencja w dokumentacji **as_is**, *argument*, [opcje|opcje]
  - Poruszanie się po man (less): */słowo , ?słowo, n, p*
  - Różne opcje wyszukiwania informacji: *–a (all pages),-f (whatis), -k(apropos)*

- opisy funkcji systemowych, np. open(2)
- man gcc

ex0_0.c

```c
#include <stdio.h>

#define KWADRAT(_v) _v*_v

int main()
{
    int v=2*KWADRAT(2);
    printf("hello world %d \n", v);

    return 0;
}
```

1. PREPROCESUJ (-E)
cpp - tworzy plik do kompilacji (-E)

gcc -E ex0_0.c > ex0_0e.c
cpp ex0_0.c > ex0_0e.c

ex0_0e.c

```c
extern void funlockfile (FILE *__stream) _
# 912 "/usr/include/stdio.h" 3 4

# 2 "ex0_0.c" 2



int main()
{
    int v=2*2*2;
    printf("hello world %d \n", v);

    return 0;
}
```

2. KOMPILUJ (-S)
cc,cc1 – kompiluje do asmblera (–S)

gcc -S ex0_0.c

ex0_0.s

```
        .file   "ex0_0.c"
        .section        .rodata
.LC0:
        .string "hello world %d \n"
        .text
.globl main
        .type   main, @function
main:
.LFB2:
        pushq   %rbp
.LCFI0:
        movq    %rsp, %rbp
.LCFI1:
        subq    $16, %rsp
.LCFI2:
        movl    $8, -4(%rbp)
        movl    -4(%rbp), %esi
        movl    $.LC0, %edi
        movl    $0, %eax
        call    printf
        movl    $0, %eax
        leave
        ret
```

3. ASEMBLER (–c)
as - tworzy kod maszynowy

ex0_0.o

***.o**

4.LINKUJ (bez opcji) **:** ld - linkuje kod z odpowiednimi bibliotekami
ldd – prezentuje biblioteki które ładuje podczas wykonywania programu

ex0_0

gcc

- GCC
  - -v (opcje kompilacji)
  - -O[**0**-3|c] (optymalizacje)
  - -shared (biblioteka) ,
  - –l / -L (linkowanie z zewnętrznymi bibliotekami)
  - -Wall (uwagi), –pedantic, -std

- **gcc** -Wall -ansi -pedantic pliki-wejściowe.c [-o plik-wykonywalny]

- Inne narzędzia:
  - gdb – debugger (opcja –g)
  - file – typ pliku
  - nm –lista symboliczna objektów - przydatne do sprawdzenia czy funkcja jest zlinkowana
  - ldd – pokazuje dynamicze biblioteki z których korzysta funkcja
  - ulimit / core dump /valgrind

make

- Makefile (domyślnie, –f <<plik>>)
  - Proces budowy kodu (przyrostowo)

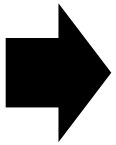
```
cel1: prerekwizyt1 prerekwizyt2 .....
<tab> <tab> komenda1
<tab> <tab> komenda2
<tab> <tab> ........

cel2:.............
<tab> <tab> ..............
```

  - zmienne: $@, $<, $?, $^
  - .PHONY – cel pozorny (nie buduje przyrostowo)
  - dependency - zależności (cel: prerekwizyty)

make

```
kompiluj:
	gcc -c  ex0_0.c   -o ex0_0.o
	gcc -o ex0_0  ex0_0.o
```

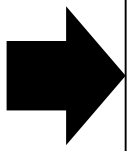


```
all: ex0_0

ex0_0 : ex0_0.o
	gcc -o ex0_0 ex0_0.c

ex0_0.o : ex0_0.c
	gcc -c ex0_0.c

clean:
	rm ex0_0.o  ex0_0
```



```
PROGRAM_NAME = ex0_0

CFLAGS = -O2 -Wall -g

ex0_0 : ex0_0.o
	gcc $(CFLAGS) $^ -o $@

%.o : %.c
	gcc $(CFLAGS) -c $<

clean:
	rm *.o
	rm  $(PROGRAM_NAME)

run: ex0_0
	@echo ""
	./program test1
```

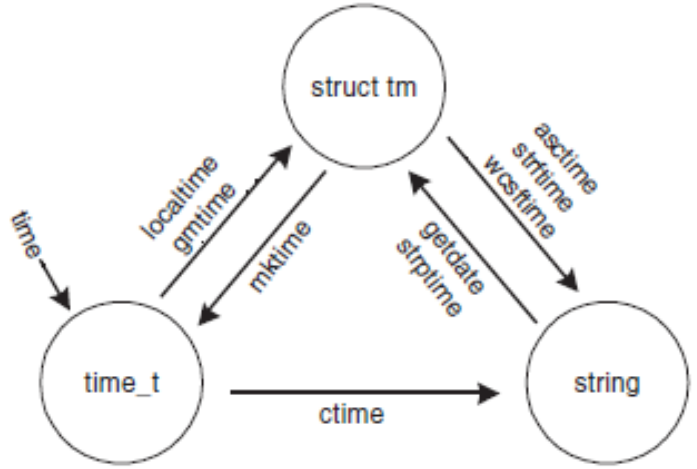
Funkcje systemowe

- Funkcja systemowa - wykonywane przez proces jądra/procesy systemowe
- Błędy
  - Charakterystyczne dla każdej funkcji
  - errno / perror / strerror
  - strerror_r - thread safe

- Kod błędu - exit
- Funkcja - atexit
- Różnica między exit a return

Data i czas w systemie UNIX

- Czas kalendarzowy:



- problem roku 2038 ☺

- Czas wykonywania programu: clock_t (CLOCKS_PER_SEC/_SC_CLK_TCK) , tms
- time() vs times() vs clock()

Standardy POSIX

- API, polecenia, powłoka
- _POSIX_VERSION
  - POSIX.1-2001 _POSIX_VERSION = 200112L
  - POSIX.1-2008 _POSIX_VERSION = 200809L

- ograniczenia systemu
- ulimit –a
- sysconf – parametry systemu
  - _SC_CHILD_MAX – ilość procesów potomnych
  - _SC_OPEN_MAX – ilość otwartych plików
  - wartość -1 (bez błędu) – oznacza brak ograniczeń

M.J.Rochkind - Programowanie w systemie UNIX dla zaawansowanych (*Advanced UNIX Programming*). Wydanie 3PL / 2EN

Proszę przeczytać rozdział –

- Rozdział 1. Pojęcia podstawowe.
- Rozdzial 2. Podstawowe operacje wejścia-wyjścia dla plików