Zestaw zadań 1

Zadanie 1.
Napisz skrypt, którego wywołanie jest następujące: $ ./skrypt01.sh <wielkość> Skrypt w podanym katalogu (i podkatalogach) wyszykuje najnowszy plik, jednocześnie większy niż podana wielkość w bajtach.

Zadanie 2.
Napisz skrypt, którego wywołanie jest następujące: $ ./skrypt02.sh Skrypt losuje liczbę całkowitą z podanego zakresu i umożliwia wykonanie tradycyjnej rozgrywki "zgadnij liczbę". Proszę sprawdzić, czy min>max. Proszę ograniczyć liczbę zgadywań.

Zadanie 3.
Napisz skrypt, którego wywołanie jest następujące: $ ./skrypt03.sh Skrypt dla zadanych wartości oblicza potęgę liczby. (jedno rozwiązanie operatorem, jedno rozwiązanie pętlą)

Zadanie 4.
Napisz skrypt, którego wywołanie jest następujące: $ ./skrypt04.sh <dokument.pdf> <podpis.png> Skrypt wykorzysta oprogramowanie ImageMagick i w pliku <dokument.pdf> nałoży zawartość pliku <podpis.png> w pozycji i . (brak możliwości zrealizowania w laboratorium uczelnianym) (uwaga do zadania: podpis rozumiec nalezy jako dowolny napis kolorem np. Niebieski, czarnym, lub innym ale z przezroczystym tlem - dlatego plik jest png)

Zadanie 5.
Napisać skrypt, który wczytuje dowolną liczbę parametrów i wypisuje je w odwrotnej kolejności. Przykład: $ ./skrypt05.sh a b 1 2 2 1 b a

Zadanie 6.
Napisz skrypty grające w "papier-nożyce-kamień": Jeden skrypt (serwer.sh) będzie "serwerem", który: Dla zadanego parametru wywołania, na przykład: $ ./serwer.sh 10 Wykona zadaną liczbę gier "papier-nożyce-kamień" (w podanym przykłądzie to będzie 10 razy) z sktyptami opisanymi dalej (gracz1.sh i gracz2.sh) poprzez opisany poniżej algorytm: Utwórz plik 'komenda.txt', a w nim zapisz komendę: 'start' Odczekaj, aż pojawią się oba pliki: los1.txt i los2.txt, a następnie: Odczekaj 0.1 [s] Skasuj plik 'komenda.txt' Pobierz z obu plików los1.txt i los2.txt zawartość Rozstrzygnij wynik i zapisz go do gra.log Skasuj pliki los1.txt i los2.txt Po wykonaniu zadanej liczby gier podaj sumaryczny wynik Do pliku 'komenda.txt' wpisz komendę: 'stop' Odczekaj 1 [s] Skasuj plik 'komenda.txt' Zakończ działanie. Dwa takie same skrypty (gracz1.sh i gracz2.sh), które będą graczami: Sprawdzają istnienie pliku 'komenda.txt' oraz jego zawartość: Jeśli jest w nim 'start' i jednocześnie nie ma pliku odpowiednio los1.txt lub los2.txt, to skrypt losuje 'papier-nożyce-kamień' i wylosowany wynik zapisuje odpowiednio w los1.txt lub los2.txt, Jeśli jest w nim 'stop', zakończ działanie. Jaki będzie wynik gry z parametrem 1000 ? Czy potrafisz zmodyfikować plik serwer.sh tak, aby sam uruchamiał graczy ? Czy spróbujesz rozbudować serwer.sh tak, aby także mógł obsługiwać dowolną liczbę graczy też podawaną jako parametr, np.: $ ./serwer.sh (nowe) Proszę uruchomić test dla 10 graczy i kolejno 10, 100 i 1000 rozgrywek, a następnie policzyć średnią i odchylenie standardowe wyników. Proszę o interpretację otrzymanych wartości.

Zadanie 8. **
Napisz skrypt odnajdujący w sposób numeryczny (nie algebraiczny, a optymalizacyjny) miejsca zerowe zadanej fukcji wielomianowej (czyli szukamy takiego x, dla którego f(x0 ==

0). Skrypt przyjmuje parametry w sposób następujący: $ ./skrypt08.sh n a b c d… n - stopień wielomianu a, b, c, … - współczynniki wielomianu Przykład: $ ./skrypt08.sh 5 2 6 -5 0 8 -3 To wielomian 5-tego stopnia: f(x) = 2 * x ^ 5 + 6 * x ^ 4 - 5 * x ^ 3 + 0 * x ^ 2 + 8 * x - 3

Zadanie 9.
Napisz skrypt, którego wywołanie jest następujące: $ ./skrypt09.sh Skrypt wypisuje elementów ciągu Fibonacciego.

Zestaw zadań 2

Zadanie 1.
Please, write a simple program that shows its PID, tip: getpid() stops using getchar() creates child and shows its pid, tip: fork() shows in child its PID and a parent PID (PPID), tip: getppid() stops using getchar() not a part of program, just run in terminal: show in OS: PID, PPID and a tree, tip: ps, htop, pidof, pstree

Zadanie 2.
Please, copy and extend a program from previous exercise: shows its PID, tip: getpid() stops using getchar() creates child and shows its pid, tip: fork() shows in child its PID and a parent PID (PPID), tip: getppid() stops using getchar() in a child process: create another child, that: shows its PID and a parent PID, show in OS: PID, PPID and a tree, tip: ps, htop, pidof, pstree

Zadanie 3.
Please, copy and extend a program from previous exercise: shows its PID, tip: getpid() stops using getchat() creates child and shows its pid, tip: fork() shows in child its PID and a parent PID (PPID), tip: getppid() stops using getchar() in a parent process: create another child, that: shows its PID and a parent PID, in a child process: create another child, that: shows its PID and a parent PID, show in OS: PID, PPID and a tree, tip: ps, htop, pidof, pstree

Zadanie 5.
Please, write a program (another.cc) that: says “Hello, I’m an another program!” stops using getchar() Please, write a program (exec.cc) that: says “Hello, I’m exec program!” stops using getchar() replaces own process and runs another.cc program, tip: exec() Please, explain using system commands like ps/htop what happen, when running exec.cc program.

Zadanie 6.
Please, write a program (system.cc) that: says “Hello, I’m system program!” stops using getchar() runs system command “ls -al /var/log/”, tip: system() Please, explain using system commands like ps/htop what happen, when running exec.cc program.

Zestaw zadań 4

Zadanie 1.
Stos i sterta Napisać program, który działa wg schematu: Z funkcji głównej wywoływane są kolejno dwie funkcje: void statyczna(); void dynamiczna(); Realizujące odpowiednio statyczny i dynamiczny przydział pamięci. W funkcji statyczna() proszę utworzyć znacznych rozmiarów zmienną lokalną, automatyczną, np.: double tablica[10^6] i zatrzymać działanie programu, np. funkcją blokującą wczytywanie znaku z klawiatury. W funkcji dynamiczna() proszę utworzyć znacznych rozmiarów zmienną lokalną, dynamiczną, np.: double *tablica = new double[10^6] i zatrzymać działanie programu, np. funkcją blokującą wczytywanie znaku z klawiatury. Przed zakończeniem działania funkcji proszę pamiętać o zwolnieniu zajmowanej pamięci. Między wywołaniami funkcji statyczna() i dynamiczna() proszę

wstrzymać działanie programu. Proszę w trakcie działania programu, w miejscach jego zatrzymania zaobserwować zajętość pamięci przez program (np. htop, /proc/[id]/smaps, itp.). Zsumuj wartości RSS w pliku /proc/[id]/smaps i odnieś do wartości RSS z htop.

Zadanie 2.
Rozrost zajętości pamięci Warsztat: (a) lista wskaźnikowa lub (b) tablica tablic. (a) Materiał do poznania: Lista wskaźnikowa / lista odsyłaczowa: AiSD-skrypt.pdf - strona 37, odniesienie do: Thomas Cormen, „Algorytmy i Struktury Danych", rozdział 11, str. 240 – 244. (b) Zadeklarować tablicę (np. 1000 komórek) wskaźników do tablic 2-wymiarowych (np. 1000x1000) typu double. Napisać program, który przy pomocy (a) lub (b) rozrasta się do pewnej, zadanej wielkości, np. tak: $ ./zadanie02 Program tworzy (a) lub (b) i oczekuje na naciśnięcie klawisza, aby zakończyć działanie. Proszę prześledzić działanie programu za pomocą: free, htop, vmstat oraz zawartość /proc Język programowania: dowolny, zalecany czysty ANSI C. Proszę dodać przed każdym utworzeniem nowej komórki (nowego elementu w (a) lub nowej tablicy w (b)) przerwę i zwrócić uwagę jak w czasie rozrasta się zajętość pamięci.

Zadanie 3.
Kontrola zajętości pamięci Zapoznać się z rozwiązaniem 'ulimit' i w taki sposób zestawić ograniczenie, aby przerwać działanie programu z zadania 2 w zadanym momencie. Czy ulimit dotyczy konta, czy pojedynczego procesu ?

Zadanie 4.
Śledzenie wykonania programu Zapoznać się z wynikiem działania programu strace, np.: $ strace date Proszę zinterpretować wynik działania. Lub napisać prosty program typu "Hello world".

Zestaw zadań 5

zadanie 1.
Badanie stopnia upakowania (kompresji) Spakuj pliki dowolnym programem (gzip, zip) i wyjaśnij nowe rozmiary plików.

zadanie 2.
Dowiązania Do pliku losowy.dat stwórz dowiązanie miękkie i twarde, odpowiedzio: losowy-soft.dat, losowy-hard.dat Korzystając z poleceń: ls, du oraz stat wyjaśnij zajętość przestrzeni przez poszczególne dowiązania.

zadanie 3.
System plików Na pliku pusty.dat załóż system plików ext4 i zamontuj w systemie plików. Podpowiedź: polecenia mkfs.ext4 oraz mount Spraw, aby montowanie/odmontowywanie tego systemu plików możliwe było przez użytkownika bez uprawnień root (podpowiedź: plik /etc/fstab).

zadanie 6.
Napisz polecenie/skrypt wykrywający wszystkie przypadki zapętlonego linkowania miękkiego. ./sprawdz-miekkie-linki.sh Podaj liczbę zapętleń (wypisz je) oraz długość poszczególnych zapętleń.

zadanie 7.

Napisz polecenie/skrypt wykrywający wszystkie przypadki hardlinkowania. ./sprawdz-twarde-linki.sh Podpowiedzi: Sprawdzenie, ile dowiązań ma plik (wpis w pliku katalogu): stat Znalezienie plików o zadanej liczbie dowiązań: find -links

zadanie 8.

Napisz polecenie/skrypt wypisujący statystykę plików w zadanym drzewie dla każdej konfiguracji ustawień dostępu. ./statystyka-uprawnien.sh rwxr-xr-x 20 rw-rw-rw- 15 …. Ile będzie takich konfiguracji ? (pozycje od lewej) Pozycja 1: typ pliku: b, c, d, p, f (-), l, s = 7 możliwości Pozycja 2: r lub (-) = 2 możliwości Pozycja 3: w lub (-) = 2 możliwości Pozycja 4: x lub S lub (-) = 3 możliwości Pozycja 5: r lub (-) = 2 możliwości Pozycja 6: w lub (-) = 2 możliwości Pozycja 7: x lub S lub (-) = 3 możliwości Pozycja 8: r lub (-) = 2 możliwości Pozycja 9: w lub (-) = 2 możliwości Pozycja 7: x lub T lub t lub (-) = 4 możliwości Liczba możliwości: 16128 Wypisać tylko te, dla których znaleziono pliki.