Systemy operacyjne

Pracownia programistyczna 3

Zadanie 1

Bez konsolidacji z biblioteką SDL (Simple DirectMedia Layer) odczytaj jej wersję przy użyciu funkcji `SDL_GetVersion`. W tym celu wykorzystaj funkcje dynamicznego konsolidatora: `dlopen`, `dlsym` i `dlclose`. Przed i po załadowaniu biblioteki uruchom program `pmap`, aby pokazać miejsce w przestrzeni adresowej procesu, gdzie konsolidator umieścił bibliotekę.

Zadanie 2

Przeczytaj rozdział 4.2 z następującego dokumentu: Memcheck: a memory error detector. Napisz program powodujący jak najwięcej usterek wspomnianych w tym dokumencie i zobacz czy rzeczywiście narzędzie `memcheck` je wykrywa.

Zadanie 3

Napisz program zliczający ilość linii w plikach tekstowych. Listę plików program ma pobrać ze standardowego wejścia. Na standardowym wyjściu dla każdego pliku ma się pojawić jego ścieżka oraz oddzielona spacją ilość linii w tym pliku. Następnie zrównoleglij swój program w taki sposób by przetwarzaniem plików zajmowała się pewna liczba procesów (ich ilość będzie pobrana z argumentów programu).

Do przechowywania stanu przetwarzania wykorzystaj współdzieloną strukturę danych między głównym procesem i procesami robotników (utworzonych wywołaniem `fork`). Do koordynacji użyj blokad współdzielonych `pthread_mutexattr_setpshared`. Do przydziału pamięci współdzielonej użyj wywołania `mmap` z flagą `MAP_ANON`.

Zadanie 4

Napisz program, który będzie przetwarzał ciąg znaków zamieniając wielkość liter. Operacja ma być wykonana w miejscu tj. operacja ma działać w obrębie jednego pliku. W pierwszym rozwiązaniu użyj buforowanych strumieni standardowej biblioteki C (funkcje `fopen`, `fread`, ...), a w drugim zmapowanych plików (wywołanie systemowe `mmap`). Zminimalizuj ilość odwołań systemowych (polecenie "`strace -c`"). Porównaj wydajność programów dla dużych plików.

Zadanie 5+

Zaimplementuj prosty menadżer pamięci przestrzeni użytkownika, składający się z czterech funkcji o nastepujących sygnaturach:

```
void *malloc(size_t size);
void *calloc(size_t count, size_t size);
void *realloc(void *ptr, size_t size);
void  free(void *ptr);
```

Znaczenie tych funkcji jest dokładnie opisane w podręcznikach systemowych. Pamięć dla procesu przydzielaj poszerzając segment danych (wywołanie `mmap` lub `sbrk`). Do zarządzania listą nieużytków wykorzystaj listę dwukierunkową ze strategią `first-fit`. Przy zwalnianiu bloków gorliwie wykonuj operację scalania.

Następnie skompiluj menadżer jako bibliotekę dzieloną. Przesłaniając symbole w/w funkcji ze standardowej biblioteki przy pomocy zmiennej środowiskowej `LD_PRELOAD` (więcej na ten temat w podręczniku `ld.so`), podmień standardowy menadżer na swój i zobacz czy działa dla prostych programów.

Zadanie 6+

Empirycznie wyznacz:

- długości linii pamięci podręcznej,
- wielkości pamięci podręcznej,
- ilości wpisów w TLB.

W tym celu musisz pobrać z systemu wielkość strony (`getpagesize`), umieć dokładnie zliczać czas trwania operacji (`gettimeofday`) oraz przydzielić pewien potencjalnie duży kawałek pamięci o adresie podzielnym przez jakąś potegę dwójki (`posix_memalign`). Skorzystaj z faktu, że aplikacja charakteryzująca się słabą lokalnością (dużo *cache miss*) zabiera wyraźnie więcej cykli procesora. Czas trwania odpowiednio spreparowanej pętli powinien umożliwić pomiar w/w wartości. Wyniki umieść na wykresie (można w tym celu użyć programu `GNU plot`).