Systemy komputerowe

Lista zadań nr 6

Na zajęcia 4 – 8 kwietnia 2019
(poprawiona)

UWAGA! Rozwiązania zadań 1–5 studenci będą prezentować przy pomocy komputera (x86-64 GNU/Linux) z projektorem dostarczonych przez prowadzącego ćwiczenia. Prezentowane rozumowania należy uzasadnić wydrukami z odpowiednich narzędzi. Można używać notatek.

Zadanie 1 (2pkty*). Rozważmy ponownie plik `swap.c` (zadanie 2, lista 5) oraz relokowalny plik `swap.o` otrzymany w wyniku kompilacji. Złącz plik `swap.o` z plikiem `main.o` otrzymanym w wyniku kompilacji pliku `main.c` zawierającego kod: `int buf[5]; int main(void) {return 0;}`.

- **Złączenie wykonaj za pomocą polecenia** `gcc -Wl,-M=main.map main.o swap.o -o main`. **Spowoduje to, oprócz stworzenia pliku wykonywalnego** `main`, **wygenerowanie pliku** `main.map`. **Z tego pliku odczytaj adres, pod którym linker umieścił sekcję** `.bss` **pliku** `swap.o`**. Adres ten będzie potrzebny przy obliczaniu niektórych relokacji w następnym punkcie zadania.**
- Odczytaj adresy które linker przyporządkował sekcjom `.text` i `.data` oraz funkcjom `incr` oraz `swap` w pliku wykonywalnym, następnie wykonaj algorytm obliczania adresów relokacji dla każdego wpisu z tablicy `.rel.text` pliku ~~`main.o`~~ `swap.o`.
- Potwierdź poprawność swoich obliczeń przez inspekcję zdezasemblowanego kodu obydwu funkcji w pliku wykonywalnym.
- Sformułuj algorytm tworzenia rekordów relokacji dla relokacji typu `R_X86_64_PC32`.

Wskazówka: CSAPP3e:§7.7

Zadanie 2. Sformułuj algorytm tworzenia rekordów relokacji dla relokacji typu `R_X86_64_32` oraz `R_X86_64_64`. Jako przykład użyj wpisu w tablicy `.data.rel` w pliku ~~`main.o`~~ `swap.o`.

Zadanie 3. Wpis w tablicy symboli dla zmiennej `buf` w pliku `swap.o` wskazuje, że zmienna ta znajduje się w sekcji `COMMON`. Zmodyfikuj deklarację zmiennej `buf` w pliku `swap.c` tak, by wpis w tablicy symboli dla `buf` wskazywał na `UNDEF`. Jaka jest rola tych wpisów w procesie tworzenia pliku wykonywalnego przez linker?

Zadanie 4. Na podstawie rozdziału §7.12 podręcznika *„Computer Systems: A Programmer's Perspective"* opisz proces **leniwego wiązania** (ang. *lazy binding*) symboli i odpowiedz na następujące pytania:

- Czym charakteryzuje się **kod umieszczalny pod dowolnym adresem** (`PIC`)? Czy w poprzednich zadaniach spotkałeś się już z takim kodem?
- Do czego służą sekcje `PLT` i `GOT` – jakie dane są tam przechowywane?
- Czemu sekcja `GOT` jest modyfikowalna, a sekcje kodu i `PLT` są tylko do odczytu?
- Co znajduje się w sekcji `.dynamic`?

Wskazówka: Wykorzystaj rysunek 7.19.

Zadanie 5. Używając narzędzi do analizy **plików relokowalnych** w formacie `ELF` i bibliotek statycznych, tj. `objdump`, `readelf` i `ar` odpowiedz na następujące pytania:

1. Ile modułów translacji zawierają biblioteki `libc.a` i `libm.a` (katalog `/usr/lib/x86_64-linux-gnu`)?
2. Czy polecenie «`gcc -Og`» generuje inny kod wykonywalny niż «`gcc -Og -g`»?

3. Z jakich bibliotek współdzielonych korzysta interpreter języka `Python` (plik `/usr/bin/python`)?

Zadanie 6. Rozważmy dysk o następujących parametrach: jeden talerz; jedna głowica; 32768 ścieżek na powierzchnię; 512 sektorów na ścieżkę; 7200 obrotów na minutę; czas wyszukiwania: 1ms na przeskoczenie o 2048 ścieżek.

- Jaki jest średni czas wyszukiwania?
- Jaki jest średni czas opóźnienia obrotowego?
- Jaki jest czas transferu sektora?
- Jaki jest całkowity średni czas obsługi żądania?

Zadanie 7. Rozważmy dysk o następujących parametrach: 360 obrotów na minutę, 512 bajtów na sektor, 96 sektorów na ścieżkę, 110 ścieżek na powierzchnię. Procesor czyta z dysku całe sektory. Dysk sygnalizuje dostępność danych zgłaszając przerwanie na każdy przeczytany bajt. Jaki procent czasu procesora będzie zużywała obsługa wejścia-wyjścia, jeśli wykonanie procedury przerwania zajmuje 2.5μs? Należy zignorować czas wyszukiwania ścieżki i sektora.

Do systemu dodajemy kontroler DMA. Przerwanie będzie generowane tylko raz po wczytaniu sektora do pamięci. Jak zmieniła się zajętość procesora?

Zadanie 8. Moduł DMA może pracować w dwóch trybach: w trybie podkradania cykli (ang. *bus stealing* lub *cycle stealing mode*), albo w trybie przesyłania porcjowego (ang. *burst mode*). W jakich zastosowaniach sprawdzi się pierwszy, a w jakich drugi z tych trybów pracy DMA? W przeważającej większości systemów implementujących moduły DMA, procesor ma niższy priorytet dostępu do pamięci głównej niż moduły DMA. Dlaczego?

Wskazówka Zakładamy, że w danym cyklu z magistrali korzystać może tylko jedno urządzenie. W trybie podkradania cykli gdy zarówno CPU jak i DMA zgłosi potrzebę przesłania danych, to CPU zostaje wstrzymany. Wtedy DMA transferuje jedno słowo po czym zwalnia magistralę. W trybie przesyłania porcjowego DMA zajmuje magistralę do czasu, aż prześle całą zawartość bufora.

Zadanie 9. Szyna systemowa i szyna pamięci w systemie komputerowym mają szerokość 32-bitów i przepustowość 10 milionów transferów na sekundę. Procesor wykonuje 32-bitowe instrukcje, z których 40% wymaga przesłania do albo z pamięci 4-bajtowych słów. O ile procent może zmienić się liczba wykonywanych instrukcji w wyniku aktywności modułu DMA, jeśli transferujemy z dysku dane z prędkością 2MB/s a moduł DMA działa w trybie podkradania cykli.