SCR LISTA4 ROZSZERZONE

Denis Firat

February 2021

Zadanie 5

Zadaniem polecenia strace jest śledzenie interakcji wywołanego programu z systemem, obserwuje on wywołania systemowe oraz sygnały procesu. Polecenie strace, działa tak długo jak wywołany program żyje. Strace przydaje sie do diganozowania pracy programu, możemy diagnozować program poprzez uruchomienie programu z strace lub "podpiać" sie do już istniejacego procesu. Chyba najważniejsza opcja jest -e, gdzie dobierając wartość możemy decydować o tym, jakie informacje śledzimy. W dokumentacji strace mamy wypisane możliwe inforamcje do śledzenia. Kolejna ciekawa opcja to -p, która pozwala nam przyczepić sie do już istniejacego procesu i go obserwować.Dokumentacja polecenia strace jest obszerna, co tylko dowodzi jego popularności przy diagnozie programów.

Zadanie 6

Do tego zadania wykorzystałem program zadanie1.c, zakomentowałem tylko cześć programu odpowiedzialna za nieskończone zapetlanie. Gdy przeprowadze doświadczenia z poleceniem strace na uproszczonej wersji, sprawdze jak zachowuje sie strace w bardziej "ekstremalnych" warunkach. Po uruchomieniu poleceniem strace z programem, można zauważyć cała serie wywołań systemowych, a dopiero na samym końcu jest polecenie write("Witam pozdrawiam").

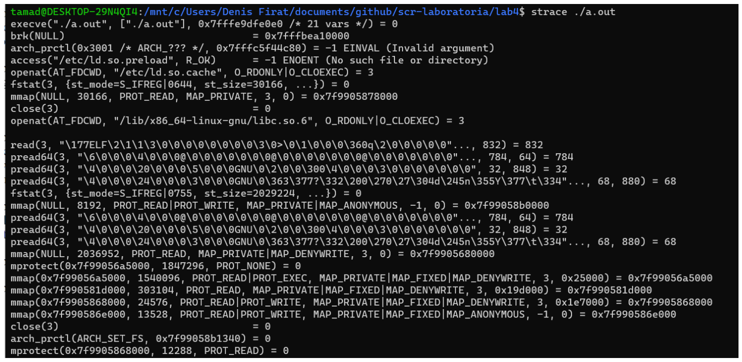

Figure 1: Odpowiedź polecenia strace przy uproszczonym programie

Najcześciej wystepujace wywołania: pread64, mmap, archprctl oraz brk. Szczerze powiedziawszy, cieżko mi zinterpretować wyniki polecenia strace, mmap służy do odwzorowywania cześci pliku w przestrzeni adresowej, pread64 zczytuje bity z deskrypora pliku, a brk zmienia rozmiar segmentacji danych. Tak jak mniej wiecej rozumiem poszczególne wywołania, tak cieżko mi prześledzić co pokolei sie dzieje podczas uruchomienia naszego programu. Ciekawiej zrobiło sie, gdy uruchomiłem program zadanie1.c z odkomentowanym blokiem nieskończonego zapetlenia. Sytuacja hipotetyczna(głównie dlatego, że kompiler ostrzegł mnie przed zapetleniem funkcji sleep oraz dlatego, że wdrożyłem to świadomie), ale załóżmy, że nie zdawałbym sobie sprawy z nieskończonej petli. Uruchomiłbym program i czekał na jego zakończenie i jak łatwo sie domyśleć, nie doczekałbym sie.

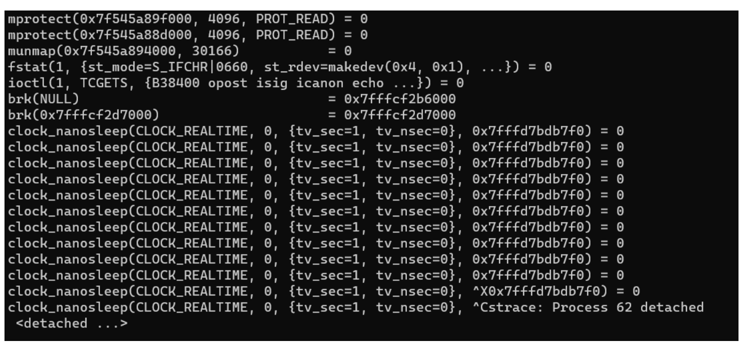

Figure 2: Odpowiedź polecenia strace przy nieskończonej petli

Na pomoc przychodzi polecenie strace, które w jasny sposób wskazuje mi źródło problemu czyli ciagłe wywoływanie clocknanosleep(), teraz wystarczy wyszukać w programie miejsca gdzie wywołuje funkcje sleep i znaleźć winowajce.

Zadanie 7

Program z listy zkompilował sie bez problemu, po uruchomieniu wyrzuca dobrze mi znany bład segmentation fault, już na tym etapie jest to dla mnie wiadomość, że prawdopodobnie gdzieś wychodzimy poza zakres przydzielonej pamieci(czesto zdarza sie ten bład przy pracy na tablicach).

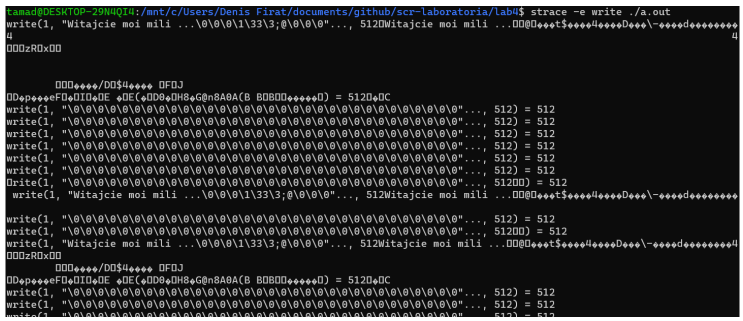

Figure 3: Odpowiedź strace na program zadanie7.c

Z pomoca polecenia strace uruchomiłem program zadanie7.c, po wstepnej obserwacji zauważyłem, że coś jest nie tak przy wywoływaniu write(). Uruchomiłem jeszcze raz program z użyciem polecenia strace, tym razem użyłem opcji -e write, która pozwoliła mi odsiać nie interesujace mnie wywołania. Wywołanie write() działało wporzadku, do momentu gdy kończyły sie trzy kropki w "Witajcie moi mili ..." ponieważ zaraz potym pojawiały sie dziwne znaki. Po sprawdzeniu jak w programie wyświetlany jest napis wszystko stało sie jasne. Brak ograniczenia na petli for sprawił, że znaki z tablicy znaków, przekazywane były nawet po przekroczeniu zakresu tablicy co powodowało pojawianie sie tych znaków. Dopiero ograniczenie wykonań petli do np. 5 sprawiło, że program zadziałał poprawnie.