Systemy operacyjne

Lista zadań nr 5

Na zajęcia 12 listopada 2020

Należy przygotować się do zajęć czytając następujące rozdziały książek:

- Tanenbaum (wydanie czwarte): 4.1, 4.2, 10.6
- APUE (wydanie trzecie): 3.15, 4.14 - 4.18, 17.2
- Linux Programming Interface: 62

UWAGA! W trakcie prezentacji należy być gotowym do zdefiniowania pojęć oznaczonych **wytłuszczoną** czcionką.

Zadanie 1. Wyjaśnij czym są **punkty montażowe**, a następnie wyświetl listę zamontowanych systemów plików. Które z nich przechowują dane w pamięci stałej komputera? Na podstawie `mount(8)` wyjaśnij znaczenie następujących atrybutów punktów montażowych: «`noatime`», «`noexec`» i «`sync`», a następnie podaj scenariusz, w którym ich zastosowanie jest pożądane.

Wskazówka: Rozważ semantykę wymienionych atrybutów w kontekście systemu plików na przenośnym dysku USB.

Zadanie 2. W systemach uniksowych katalog to ciąg bajtów reprezentujący listy rekordów `dirent(3)`. Na podstawie §10.6.3 (rysunek 10-32) przedstaw reprezentację katalogu, a następnie wyjaśnij jak przebiegają operacje usuwania i dodawania pliku. W pierwszym przypadku rozważ scenariusz, w którym w reprezentacji katalogu za lub przed usuwanym wpisem istnieją **nieużytki**. W drugim, kiedy w pliku katalogu nie udaje się znaleźć wystarczająco dużo miejsca na przechowanie wpisu. Jądro leniwie wykonuje operację **kompaktowania** na katalogach – kiedy opłaca się ją zrobić?

Zadanie 3. Korzystając z poleceń «`stat`» i «`ls -lia`» zaprezentuj jak jądro systemu operacyjnego trawersuje **ścieżkę bezwzględną** «`/usr/share/man/man1/ls.1`». Od jakiego numeru **i-węzła** algorytm zaczyna działanie? Skąd sterownik uniksowego systemu plików wie gdzie na dysku znajduje się i-ty bajt pliku? Próba utworzenia dowiązania do pliku «`/proc/version`» kończy się błędem «`EXDEV`». Czemu nie możemy tworzyć dowiązań do plików znajdujących się w obrębie innych zamontowanych systemów plików?

Zadanie 4. W bieżącej wersji biblioteki «`libcsapp`» znajduje się plik «`terminal.c`». Przeczytaj pierwsze dwa podrozdziały §62, a następnie zreferuj działanie procedury «`tty_curpos`» odczytującej pozycję kursora terminala. Do czego służy kod sterujący «`CPR`» opisany w Terminal output sequences[1]? Posiłkując się `tty_ioctl(4)` wytłumacz semantykę rozkazów «`TCGETS`» i «`TCSETSW`», wykorzystywanych odpowiednio przez `tcgetattr(3)` i `tcsetattr(3)`, oraz «`TIOCINQ`» i «`TIOCSTI`». Na podstawie `termios(4)` wyjaśnij jak flagi «`ECHO`», «`ICANON`», «`CREAD`» wpływają na działanie **sterownika terminala**.

[1] `https://en.wikipedia.org/wiki/ANSI_escape_code#Terminal_output_sequences`

Ściągnij ze strony przedmiotu archiwum «so20_lista_5.tar.gz», następnie rozpakuj i zapoznaj się z dostarczonymi plikami.

UWAGA! Można modyfikować tylko te fragmenty programów, które zostały oznaczone w komentarzu napisem «TODO».

Zadanie 5. Uruchom program «mkholes», a następnie odczytaj **metadane** pliku «holes.bin» przy pomocy polecenia stat(1). Wszystkie pola struktury «stat» są opisane w stat(2). Oblicz faktyczną objętość pliku na podstawie liczby używanych bloków «st_blocks» i rozmiaru pojedynczego bloku «st_blksize» systemu pliku. Czemu liczba używanych bloków jest mniejsza od tej wynikającej z objętości pliku z pola «st_size»? Czemu jest większa od liczby faktycznie używanych bloków zgłaszanych przez «mkholes»?

Wskazówka: O dziurach w plikach (ang. *holes*) można przeczytać w rozdziale 3.6 APUE.

Zadanie 6. Program «listdir» wypisuje zawartość katalogu w formacie przypominającym wyjście polecenia «ls -l». Poniżej można znaleźć przykładowy wydruk, na którym widnieją odpowiednio: plik zwykły, dowiązanie symboliczne, urządzenie znakowe, plik wykonywalny z bitem set-uid, jeden katalog z ustawionym bitem set-gid i drugi z bitem sticky.

```
1 -rw-r--r--   1  cahir  cahir   2964  Fri Nov 15 14:36:59 2019  listdir.c
2 lrwxrwxrwx   1  cahir  cahir     17  Mon Nov  4 11:14:49 2019  libcsapp -> ../csapp/libcsapp
3 crw--w----   1  cahir    tty  4,  2  Tue Nov 12 08:42:33 2019  tty2
4 -rwsr-xr-x   1   root   root  63736  Fri Jul 27 10:07:37 2018  passwd
5 drwxrwsr-x  10   root  staff   4096  Mon Jan  9 13:49:40 2017  local
6 drwxrwxrwt  23   root   root  12288  Fri Nov 15 16:01:16 2019  tmp
```

Uzupełnij kod programu według wskazówek zawartych w komentarzach w kodzie źródłowym. Należy użyć:

- fstatat(2) do przeczytania metadanych pliku,
- major(3) i minor(3) do zdekodowania **numeru urządzenia**,
- readlinkat(2) to przeczytania ścieżki zawartej w dowiązaniu symbolicznym.

Implementacja iterowania zawartości katalogu będzie wymagała zapoznania się ze strukturą «linux_dirent» opisaną w podręczniku getdents(2). Wywołanie systemowe «getdents» nie jest eksportowane przez bibliotekę standardową, zatem należało je wywołać pośrednio – zobacz plik «libcsapp/Getdents.c».

Zadanie 7 (2). (Pomysłodawcą zadania jest Tomasz Wierzbicki.)

Program «mergesort» odczytuje ze standardowego wejście liczbę naturalną n, po czym czyta n liczb całkowitych. Program realizuje algorytm sortowania przez scalanie. Proces główny zajmuje się wczytywaniem danych wejściowych i drukowaniem posortowanego ciągu. Żeby posortować liczby, program uruchamia podproces, który wykonuje procedurę «Sort». Rozmawia z nim przy pomocy gniazda domeny uniksowej unix(7), które tworzy z użyciem socketpair(2), czyli **lokalnej dwukierunkowej** metody komunikacji międzyprocesowej. Jeśli proces sortujący otrzyma od rodzica pojedynczą liczbę, to natychmiast odsyła ją swojemu rodzicowi i kończy działanie. Jeśli dostanie więcej liczb, to startuje odpowiednio lewe i prawe dziecko, po czym za pomocą procedury «SendElem» przesyła im liczby do posortowania. Następnie wywołuje procedurę «Merge», która odbiera od potomków posortowane ciągi, scala je i wysyła do procesu nadrzędnego.

Twoim zadaniem jest uzupełnienie procedury «Sort» tak by wystartowała procesy potomne i uruchomiła procedury «SendElem» i «Merge». Należy odpowiednio połączyć procesy z użyciem gniazd oraz zamknąć niepotrzebne gniazda w poszczególnych procesach. Posługując się rysunkiem wyjaśnij strukturę programu. Kiedy tworzysz podprocesy i gniazda? Kiedy zamykasz niepotrzebne gniazda? Jak wygląda przepływ danych?

Skrypt «gen-nums.py» przyjmuje w linii poleceń n, czyli liczbę elementów do wygenerowania. Po uruchomieniu drukuje n na standardowe wyjście, po czym drukuje n losowych liczb całkowitych. Produkowane dane są w odpowiednim formacie do wprowadzenia do programu «mergesort».

UWAGA! Wszystkie procesy muszą działać w stałej pamięci!