Zaawansowane systemy operacyjne

dr inż. Zbigniew Suski

zsuski@pjwstk.edu.pl

ftp.pjwstk.edu.pl\zsuski\zso

Literatura

- Silberschatz A. Galvin P.B.: *Podstawy systemów operacyjnych.* WNT 2000.
- Stevens W.R.: *UNIX - Programowanie usług sieciowych Tom 1 - API: gniazda i XTI.* WNT 2000.
- Stevens W.R.: *UNIX - Programowanie usług sieciowych Tom 2 - komunikacja międzyprocesowa.* WNT 2001.
- Coulouris G. Dollimore J. Kindberg T.: *Systemy rozproszone, podstawy i projektowanie.* WNT 1999.
- Tanenbaum A.S.: *Rozproszone systemy operacyjne.* PWN 1997.

Proces

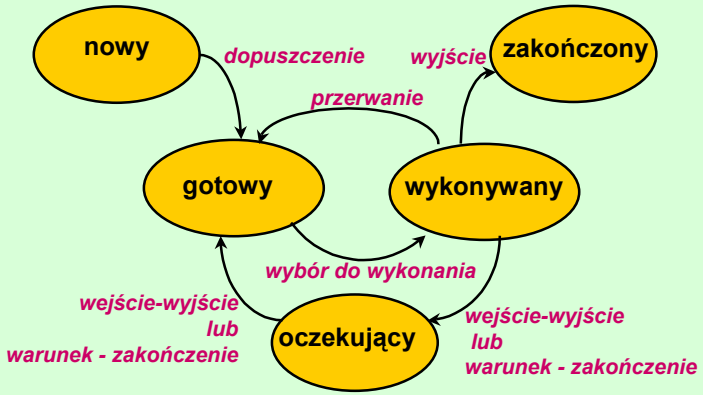
Proces – to wykonywany program. Zwykle potrzebuje pewnych zasobów, takich jak:

- czas procesora,
- pamięć operacyjna,
- pliki,
- urządzenia we/wy.

Program jest obiektem pasywnym

Proces jest obiektem aktywnym

Stany procesu



Dlaczego procesy współbieżne?

- podział zasobów fizycznych (*ze względu na ich ograniczoność*),
- podział zasobów logicznych (*ze względu na konieczność wspólnego dostępu do niektórych zasobów*),
- przyśpieszanie obliczeń (*równoległe wykonywanie zadań*),
- modularność,
- wygoda *(jednoczesne wykonywanie wielu zadań przez jednego użytkownika).*

Proces niezależny

Nie może wpływać na zachowanie innych procesów, ani inne procesy nie mogą oddziaływać na niego.

WŁAŚCIWOŚCI:

- na jego stan nie wpływa w żaden sposób, żaden inny proces,
- wynik jego pracy zależy wyłącznie od stanu wejściowego,
- jego działanie jest powielarne,
- jego wykonanie może być wstrzymywane i wznawiane bez żadnych szkodliwych skutków.

Proces współpracujący

Może wpływać na zachowanie innych procesów, lub sam może ulegać wpływom innych procesów.

WŁAŚCIWOŚCI:

- jego stan jest dzielony wraz z innymi procesami,
- nie da się określić z góry wyniku jego działania, gdyż zależy on od względnej kolejności wykonywania procesu,
- wynik działania może nie być zawsze taki sam, przy takich samych danych wejściowych.

Jeżeli proces dzieli jakiekolwiek dane z innymi procesami, to jest procesem współpracującym.

Procesy – podstawowe zagadnienia

- Podział czasu
- Jednoczesność
- Współpraca i współzawodniczenie
- Komunikacja i synchronizacja
- Wzajemne wykluczanie

```
do {
. . . .
sekcja_wejściowa
sekcja_krytyczna
sekcja_wyjściowa
reszta
} while (1);
```

Procesy – podstawowe zagadnienia

Bezpieczeństwo:

- Dla sekwencyjnych:
  - własność częściowej poprawności
  - własność stopu
- Dla współbieżnych:
  - własność bezpieczeństwa
  - własność żywotności
  - własność sprawiedliwości

Zakleszczenie

Zagłodzenie

Problem producenta i konsumenta

Przekazywanie parametrów do procesu

```
main(argc, argv, envp)
int argc;          /* liczba napisów będących argumentami */
char *argv[ ];     /* tablica wskaźników do napisów */
                   /* pierwszy napis to nazwa polecenia
char *envp[ ];     /* tablica wskaźników do napisów
                   określających zmienne środowiskowe ; np
                   SHELL=/bin/sh */
{
...
}

extern char **environ;

char *ptr;
        ptr = getenv("HOME");
        printf("HOME=%s\n",ptr);
```


Funkcja *fork*

- Przydzielenie nowemu procesowi pozycji w tablicy procesów.
- Przydzielenie procesowi potomnemu identyfikatora.
- Utworzenie logicznej kopii kontekstu procesu macierzystego.
- Zwiększenie plikom związanym z procesem liczników w tablicy plików oraz i-węzłów.
- Przekazanie wartości 0 procesowi potomnemu i identyfikatora procesu potomnego procesowi macierzystemu.

Funkcja *fork* i *exec*

```
if ((f=fork())==0)   {
        execl("proces_1", "p1", NULL);
        return(0);  }
  else    if (f == -1)    {
                printf("\nBLAD \n");
                return(1); }
        else     if ((f=fork())==0)      {
                        execl("proces_2", "p2", NULL);
                        return(0);  }
                else    if (f == -1)      {
                        printf("\nBlad 2 \n");
                        return(1); }
                        else       {
                                execl ("proces_3", "p3", NULL);
                                return(0); }
. . . .
```

Pobieranie niektórych atrybutów procesu

- Identyfikator procesu **getpid()**;
- Identyfikator procesu macierzystego **getppid()**;
- Rzeczywisty identyfikator użytkownika **getuid()**;
- Rzeczywisty identyfikator grupy **getgid()**;
- Efektywny identyfikator użytkownika **geteuid()**;
- Efektywny identyfikator grupy **getegid()**;
- Identyfikator grupy procesów **getpgrp()**;