5 1992

informatyka

WYDAWNICTWO SIGMA NOT

Cena 18 000 zł

informatyka

nr 5 1992
Maj
Rok wydania XXVII

Miesięcznik
ISSN 0542-9951
INDEKS 36124

WYDAWCA:
Wydawnictwo Czasopism i Książek Technicznych SIGMA NOT
Spółka z o.o.
ul. Biała 4
00-950 WARSZAWA
skrytka pocztowa 1004

Redakcja:
01-552 Warszawa,
Pl. Inwalidów 10, p. 104, 105
tel. 39-14-34

Materiałów nie zamówionych redakcja nie zwraca

W numerze:

W najbliższych numerach:

- **Zdzisław Wrzeszcz zajmuje się metodologią gromadzenia i przetwarzania informacji faktograficznej (informacji FG).**
- **J. Piecha, A. Przybylski i J. Zyguła charakteryzują współczesne narzędzie symulacji – Authorware Professional for Windows – ułatwiające projektowanie materiałów wspomagających nauczanie.**
- **Jerzy R. Chrząszcz i Grzegorz B. Mazur opisują laboratoryjny system mikroprocesorowy – istotne narzędzie pracy dydaktycznej, doświadczalnej i konstrukcyjnej w dziedzinie budowy systemów.**
- **Zdzisław Szyjewski omawia systemy otwarte, gwarantujące możliwość łatwego przenoszenia systemu informatycznego.**

Warunki prenumeraty

Przyjęcie prenumeraty – wyłącznie na podstawie dokonanej wpłaty na drukach dostarczanych dotychczasowym prenumeratorom przez Wydawnictwo, lub nowym – po uprzednim zgłoszeniu zapotrzebowania (pisemnie lub telefonicznie) w Zakładzie Kolportażu Wydawnictwa.
Blankiet wpłaty – powinien zawierać następujące informacje: dokładna nazwa i adres (z kodem pocztowym) zamawiającego, tytuły zamawianych czasopism, ich liczbę i okres prenumeraty.
Wpłata – zgodnie z podanymi cenami należy dokonać w banku lub w UPT na konto podane na naszym blankiecie, tj:
Państwowy Bank Kredytowy III O/Warszawa nr: 370015-1573-139-11
Prenumeratorzy zbiorowi – osoby prawne obowiązują blankiety „Wpłata-Zamówienie". Cena normalna.
Prenumeratorzy indywidualni – osoby fizyczne obowiązują blankiety typu przekazy dla wpłat na rachunki bankowe. Cena normalna.
Prenumerata ulgowa – zgodnie z podaną ceną ulgową przysługuje wyłącznie osobom fizycznym, będącym członkami SNT, studentom i uczniom szkół zawodowych. Uczniowie szkół ogólnokształcących mogą zamówić w prenumeracie ulgowej tylko miesięcznik „Aura"
Uwaga! w podanym okresie prenumeraty można zamówić tylko po jednym egzemplarzu z każdego tytułu.
Prenumerata ze zleceniem wysyłki za granicę – cena prenumeraty ze zleceniem wysyłki za granicę jest dwukrotnie wyższa od ceny normalnej.
Należy podać dokładny adres odbiorcy za granicą.
Terminy przyjmowania prenumeraty:
– do 10 listopada na I, II, III, IV kwartał następnego roku
– do 28 lutego na II, III, IV kwartał br.
– do 31 maja na III i IV kwartał br.
– do 31 sierpnia na IV kwartał br.
Zmiany w prenumeracie, np. zmiana liczby tytułów, liczby egzemplarzy, rezygnacja z prenumeraty, można zgłaszać tylko w podanych terminach z mocą obowiązującą od następnego kwartału.
Egzemplarze archiwalne (z lat ubiegłych)
Można nabyć za gotówkę w Klubie Prasy Technicznej, Warszawa, ul. Mazowiecka 12 (tel. 26-80-16) lub zamówić pisemnie w Zakładzie Kolportażu, Dział Handlowy, 00-950 Warszawa, skr. poczt. 1004 (tel. 40-37-31), na rachunek lub za zaliczeniem pocztowym.
Informacji o prenumeracie udziela: Zakład Kolportażu Wydawnictwa SIGMA-NOT Spółka z o.o., 00-716 Warszawa, ul. Bartycka 20, skr. 1004. Telefony: 40-00-21 wewn. 293, 295, 299 lub 40-30-86, 40-35-89.
Wstępna cena jednego egzemplarza na 1992 rok: normalna – 18 000 zł, ulgowa – 13 500zł
Wartość prenumeraty (w zł):
Normalna: kwartalna – 54 000, półroczna 108 000, roczna 216 000
Ulgowa: kwartalna – 40 500, półroczna 81 000, roczna 162 000
Uwaga: W przypadku zmiany cen w okresie objętym prenumeratą, prenumeratorzy zobowiązani są do dopłaty różnicy cen.

Skład i druk: Drukarnia SIGMA NOT Sp. z o.o. z. 141/92

PIOTR KOWALSKI
Instytut Badań Systemowych PAN
Warszawa

T^3 – program przetwarzania wielojęzycznych tekstów naukowo-technicznych

W artykule podano krótką charakterystykę dwu głównych nurtów – WYSIWYG i ML – oprogramowania do przetwarzania tekstów technicznych (TWP) wraz z przykładami oraz omówiono bliżej, ceniony w Europie Zachodniej i USA, program T^3 dla komputerów klasy IBM PC zaliczany do grupy WYSIWYG TWP. Scharakteryzowane zostały zasady posługiwania się programem T^3, jego główne elementy użytkowe, przygotowywanie dokumentów i atrybuty użytkowników. Zaprezentowano podejście do przetwarzania tekstów w języku polskim z użyciem tego programu, możliwość współpracy z T_EXem i porównanie z ChiWriterem.

Programy TWP

Komputerowe przetwarzanie trudnych tekstów naukowo-technicznych, zwłaszcza o charakterze matematycznym (ang. *mathematical text processing*), jest wykonywane przy użyciu coraz bardziej rozbudowywanego oprogramowania noszącego w literaturze angielskojęzycznej wspólną nazwę TWP (ang. *technical word processors*) – programów przetwarzania tekstów technicznych. Korzystanie z takiego oprogramowania stało się warunkiem koniecznym efektywnej wymiany informacji w zainteresowanych środowiskach.

Szczególne utrudnienia, nieodzownie związane z przygotowywaniem dokumentów matematycznych, są konsekwencją [10] znacznie większej złożoności logicznej tekstów matematycznych w porównaniu do tekstów innego rodzaju. Odpowiednio bardziej skomplikowane są więc – uformowane przez lata prób i gromadzenia doświadczeń – sztuka i nauka projektowania struktury i szaty graficznej publikacji matematycznych w taki sposób, żeby korzystanie z nich było jak najbardziej efektywne.

Złożoność dokumentów matematycznych zwiększa się szybciej niż liniowo wraz ze wzrostem ich objętości, ze względu na silne wzajemne powiązania między poszczególnymi ich częściami. Matematyka posługuje się znacznie większym alfabetem symboli podstawowych, dodatkowymi kategoriami słów (np. zmienne, operatory, indeksy) i zdań (formuły, równania). W uzupełnieniu zwykłych akapitów tekstu, pojawiają się ponadto definicje, twierdzenia lub dowody. Dla wszystkich wspomnianych tutaj dodatkowych kategorii istnieją specjalne reguły formatowania, konwencje stylistyczne i systemy numeracji ułatwiające wzajemne odwoływanie się.

Ogromne wymagania wobec programów TWP wynikają więc ze złożoności przetwarzanych tekstów i obejmują (jako zupełne minimum) korzystanie ze specjalnych zestawów symboli (np. matematycznych, chemicznych, greckich), budowanie z ich użyciem skomplikowanych wzorów i diagramów oraz dostępność zróżnicowanych krojów i wielkości pisma.

Z dogłębną analizą wymagań stawianych programom TWP można zapoznać się w części I bardzo dobrego opracowania [6]. Analiza ta obejmuje szczegółowe rozpatrzenie czterech głównych aspektów decydujących o wyborze programu TWP przez użytkownika, tj. możliwości: konstruowania wzorów (ang. *equation capabiblities*), zarządzania strukturą dokumentu (ang. *manuscript/organization capabilities*) i kontroli jego szaty graficznej (ang. *layout capabilities*) oraz sposobu współpracy z programem (ang. *user interface*) pozwalającego na wykorzystanie tych możliwości, jak również szeroko rozumianych kosztów użytkowania oprogramowania TWP oraz przewidywanych trendów w jego rozwoju.

Wśród oprogramowania z grupy TWP wyróżnia się dwa główne typy interfejsu między użytkownikiem a programem. Pierwszym z nich jest interfejs typu ML, w którym dokument przygotowywany przez użytkownika stanowi połączenie tekstu właściwego i poleceń specjalnego języka (ang. markup language) instruującego program, jak należy formatować lub składać tekst. Formatery typu ML, działając w trybie wsadowym, przetwarzają dokument na pewną zakodowaną postać, przesyłaną później do urządzeń zewnętrznych. W przypadku formaterów najwyższej klasy, wygenerowany kod jest niezależny od urządzeń zewnętrznych, co ułatwia przygotowanie całej gamy programów (ang. *drivers*) obsługi urządzeń wyjściowych (monitory graficzne, drukarki, urządzenia składu drukarskiego).

Drugi typ interfejsu nosi nazwę WYSIWYG (ang. what you see is what you get), czyli co widzisz (na ekranie) to otrzymasz (na wydruku), i oznacza taki rodzaj współpracy z programem, w której wyświetlany na ekranie obraz przygotowywanego dokumentu jest aktualizowany na bieżąco podczas redagowania i formatowania wykonywanego przez użytkownika oraz odpowiada dokładnie (lub z dobrym przybliżeniem, którego zmniejszana jakość wynika m.in. z mniejszej rozdzielczości ekranu w porównaniu do drukarki) postaci dokumentu otrzymywanej na wydruku.

Konstruowanie wzorów (WYSIWYG) odbywa się zwykle w trybie „malowania" ich na ekranie (ang. *paint-it*). Symbole składające się na wzór są bepośrednio wpisywane w odpowiednie miejsce na ekranie z użyciem określonych funkcji sterujących (klawiatura, mysz) pozwalających na: pozycjonowanie

Mgr inż. PIOTR KOWALSKI ukończył w 1985 r. Wydział Fizyki Technicznej i Matematyki Stosowanej Politechniki Warszawskiej – specjalność Matematyka Stosowana. Od 1986 r. pracuje w Instytucie Badań Systemowych PAN. Aktualne zainteresowania obejmują problematykę szeroko rozumianych obliczeń naukowych i oprogramowania wykorzystywanego do ich realizacji.

kursowa, dostęp do różnorodnych zestawów znaków, zmianę wielkości lub kroju pisma i nadawanie atrybutów. W niektórych implementacjach są przewidziane specjalne tryby wprowadzania (np. automatycznie centrujące ułamki lub dopasowujące wielkości symboli całek, sum i nawiasów) lub redagowania – uwzględniające wszystkie warstwy wielopoziomowego wiersza jednocześnie lub każdą jego warstwę składową (ew. kilka warstw) oddzielnie. Dopuszczalne jest również [3] budowanie wzorów w trybie wydawania poleceń (ang. *say-it*), których składnia przypomina język ML, ale ich efekt jest natychmiast odwzorowywany na ekranie.

Wśród programów z grupy TWP spotyka się także interfejs mieszany (hybrydowy). Niektóre programy, zwykle zaliczane do kategorii WYSIWYG, wykonują bardziej złożone operacje organizacyjne dokumentu (np. automatyczne tworzenie spisu treści, skorowidza czy numerowanie wzorów) korzystając z języka ML. Hybrydy akceptujące wprowadzanie wzorów w konwencji ML oferują możliwość ich podglądu na ekranie w odzielnym oknie. Istnieją programy, w których faza redagowania i formatowania dokumentu odbywa się w trybie WYSIWYG, ale tworzony dokument źródłowy jest przechowywany jako klasyczny dokument z poleceniami ML i na podstawie tej postaci jest drukowany. Najczęściej występującym pomostem między ideałami: wejście-WYSIWYG, wyjście-ML, są konwertery dokumentów utworzonych przez program typu WYSIWYG na dokumenty akceptowane przez program typu ML. Dwa główne typy interfejsu z użytownikiem mają swoich zwolenników i przeciwników.

Aspekt wyboru interfejsu, w odniesieniu do programów TWP, jest traktowany [3, 6] jako wyjątkowo ważny, lecz bardzo subiektywny, zależny od indywidualnych preferencji użytkownika – i jako jedyny z czterech głównych aspektów, wymykający się ocenie ilościowej. Poniżej przedstawiono zaledwie kilka poglądów na temat wyboru interfejsu ML czy WYSIWYG.

Dla użytkownika rezygnującego całkowicie z papieru i ołówka na rzecz programu TWP i sprzętu komputerowego, bardzo ważna jest możliwość „myślenia przy ekranie". Osoby, których proces myślenia jest w istotny sposób stymulowany obrazami, wybiorą prawdopodobnie interfejs WYSIWYG, pozostali będą w pełni zadowoleni lub nawet będą preferowali interfejs ML.

Programy WYSIWYG są zwykle prostsze w opanowaniu niż języki ML. Łatwość i wygoda użytkowania jest już bardziej zależna od preferencji użytkownika. Interfejs ML może być wyjątkowo frustrujący, gdy pojawia się konieczność poprawiania np. błędów składniowych, WYSIWYG – szczególnie pracochłonny, gdyż odpowiedzialność za detale formatowania złożonego tekstu spoczywa prawie całkowicie na użytkowniku. Korzystanie z odpowiednich programów TWP z grupy ML staje się niezbędne, gdy trzeba zapewnić dokumentom końcowym jakość odpowiadającą profesjonalnemu składowi drukarskiemu tekstów matematycznych.

W środowisku matematycznym [10] znany był pogląd, że zasadniczym kryterium swobodnego wyboru (nie ograniczonego np. wymaganiami wydawców prac matematycznych) między T_EX-em – najwyżej cenionym wśród matematyków programem ML – a programem typu WYSIWYG jest złożoność przygotowywanego dokumentu. To znaczy duże, skomplikowane dokumenty, takie jak książki lub prace doktorskie są opracowywane z użyciem T_EX-a, pozostałe, mniejsze i prostsze, z pomocą programu WYSIWYG. Zmodyfikowany pogląd na tę sprawę mówi, że decyduje nie złożoność dokumentu, lecz osoba wprowadzająca tekst do komputera. Matematycy piszący przy komputerze własne teksty będą zwykle preferowali T_EX-a, a personel techniczny, wykonujący za nich tę pracę, zazwyczaj programy typu WYSIWYG. Decyduje o tym zrozumienie logicznej struktury wpisywanych formuł, poczucie naturalności, łatwości nauczenia się, użycia i zapamiętania u tych pierwszych oraz brak kilku lub wszystkich tych cech u drugich.

Czytelnika zainteresowanego drobiazgową analizą względnych zalet i wad interfejsu ML w zestawieniu z WYSIWYG odsyłamy do [6 – sect. IV].

Wśród procesorów tekstów technicznych z grupy ML, T_EX jest uznawany [5] za program najbardziej dopracowany, o największych możliwościach składania tekstów o bardzo dobrej jakości. T_EX, działający pod kontrolą wielu systemów operacyjnych, jest najczęściej używany z dodatkowymi pakietami makrodefinicji (np. AMS-T_EX, LaT_EX) stanowiącymi zwykle rozszerzenie lub modyfikację tzw. wersji bazowej (ang. *plain*) – w polskiej literaturze matematycznej różnorodne aspekty systemu T_EX są dokładniej omówione w pracy [2]. Drugie miejsce w zestawieniu programów ML zajmuje program **nroff/troff** z bibliotekami makrodefinicji, pre- i postprocesorami (np. AT&T Documenter's Workbench [1]: **ditroff** – *device independent* **troff, nroff; eqn** i **neqn** – wzory matematyczne i notacja naukowa, **tbl** – tabelki, **pic** – diagramy, **grap** – rysunki i wykresy) działający w środowisku systemu UNIX.

Dla komputerów klasy IBM PC (system DOS) dużym uznaniem w grupie WYSIWYG cieszą się od dłuższego czasu programy **T³**, **ChiWriter** i **EXP** [8, 6]. Pojawiają się również nowe programy, takie jak **AmiPro** 2.0 [4], które działając w środowisku graficznym MS-Windows 3.0, uwzględniają wymagania oprogramowania TWP.

Środowisko graficzne komputerów Macintosh stanowi naturalną bazę dla implementacji programów TWP wykorzystujących interfejs WYSIWYG. Wysoko są oceniane [10] walory programów WYSIWYG – do konstruowania wzorów matematycznych – takich jak **Expressionist** i **MathType,** czy kompletnych programów TWP, takich jak **MathWriter** [12].

Przykładami rozwiązań hybrydowych są: **MS-Word** dla Macintosha [11, 17] – procesor tekstów działający w trybie WYSIWYG z wbudowanym językiem typu ML do składania wzorów matematycznych wraz z możliwością ich podglądu w oddzielnym oknie, czy **EXACT** dla IBM PC [6] – samodzielny, rezydujący w pamięci operacyjnej produkt, współpracujący z wieloma klasycznymi procesorami tekstu (np. **WordStar, PC Write, Volkswriter**), pozwalający na konstruowanie wzorów czy używanie notacji naukowej z użyciem własnego języka ML, z podglądem na ekranie i obsługą wydruku.

Inną podgrupę hybrydowych programów TWP stanowią **Leo** [8] i **SweeT_EX** [14], pozwalające na wprowadzanie kodu T_EX-owego z użyciem interfejsu WYSIWYG, oraz mniej znany produkt VorT_EX (ang. *Visually Oriented* T_EX) [6], który na stacji roboczej Sun udostępnia użytkownikowi dwa sprzężone wzajemnie okna, z podglądem graficznym oraz źródłem T_EX-owym, w taki sposób, że redagowanie w oknie podglądu jest natychmiast odwzorowywane w oknie T_EX-a i na odwrót.

Na koniec tego króciutkiego przeglądu warto jeszcze wspomnieć o konwerterach dokumentów trzech wspomnianych programów WYSIWYG (dla IBM PC) [8] na postać źródłową T_EX-a i analogicznych możliwościach programów dla Macintosha (np. Expressionist).

Ogólna charakterystyka programu T³

Wśród programów z grupy WYSIWYG, dla komputerów klasy IBM PC, dużym uznaniem w USA i Europie Zachodniej

cieszy się wspomniany już T^3 (ang. *tee cubed*) – program do przetwarzania tekstów technicznych amerykańskiej firmy TCI Software Research, Inc. Jego funkcjonalnym odpowiednikiem jest dobrze znany w Polsce ChiWriter.

T^3 istniejący na rynku od 1984 roku, zyskał w tym czasie m.in. uznanie u czytelników rubryki „Mathematical Text Processing" czasopisma *Notices of American Mathematical Society* [9], wyróżnienia czasopisma *PC Magazine* [13, 15] za wersje 2.11 (1986 r.) i 2.21 (1988 r.) oraz wysoką ocenę Bostońskiego Towarzystwa Komputerowego w dwukrotnie przeprowadzonym przeglądzie narzędzi TWP [3, 6].

Z racji swoich walorów, zdaniem autora tego artykułu, program T^3 jest wart przybliżenia go polskim użytkownikom komputerów PC.

Najważniejsze właściwości programu T^3 (wersja 2.21 z 1988 r.) są następujące:

- jest programem zaliczanym do grupy WYSIWYG (i działa wyłącznie w trybie graficznym sterownika graficznego);
- umożliwia przetwarzanie tekstów wielojęzycznych (angielskich, rosyjskich, polskich, niemieckich, skandynawskich i in.);
- daje możliwość wygodnego konstruowania wielopoziomowych wzorów matematycznych i chemicznych;
- spełnia, poza wszystkim innym, funkcję typowego procesora tekstów nietechnicznych, obsługiwanego na zasadzie wyboru z menu oraz korzystania z licznych, logicznie dobranych operacji klawiszami sterującymi;
- zawiera rozbudowany zestaw fontów tekstowych (m.in. z polskimi literami) i specjalistycznych (13 podstawowych zestawów znaków, łącznie w 40 odmianach wielkości i kroju) oraz wbudowany edytor do tworzenia i modyfikowania fontów; fonty tekstowe są dostępne w postaci proporcjonalnej lub o jednakowej szerokości; każdemu znakowi wprowadzonemu do dokumentu – niezależnie od tego, do którego fontu należy – można nadać dodatkowe atrybuty (wytłuszczenie, podkreślenie, przekreślenie lub ich kombinację);
- udostępnia starannie opracowaną metodę obsługi klawiatury (składanie, ang. *key-chording*) oraz aparat sekwencji klawiszowych (bezparametrowych klawiszowych makrodefinicji), pozwalający zapamiętać i odtworzyć wykonanie ciągu operacji;
- treść dokumentu jest zazwyczaj przygotowywana z użyciem wcześniej zdefiniowanych klawiatur; znaki przypisane klawiszom alfanumerycznym (lub ich dwuelementowym kombinacjom) tej samej klawiatury mogą należeć do różnych fontów; istnieją ponadto trzy inne metody wprowadzania znaków i cztery metody nadawania im atrybutów;
- pozwala na pisanie tekstów w długich wierszach ($\leqslant 158$ znaków), tekstów w wyróżnionych prostokątnych obszarach na stronie (np. tekstów wielołamowych) i zawierających zmienne pola wypełniane podczas wydruku (ang. *merge-print*);
- w dokumencie mogą być umieszczone znaczniki obiektów graficznych wiążące prostokątne obszary na stronie lub w wierszu, z plikiem przechowującym obraz graficzny (w postaci przeznaczonej dla drukarek laserowych lub Postcriptowych); obrazy graficzne nie są widoczne na ekranie, ale mogą być skalowane,
- zapewnia automatyczne formatowanie dokumentu na podstawie wzorców określających postać wiersza (ang. *line format*) i strony (ang. *page format*);
- umożliwia kontrolę pisowni wyrazów języka angielskiego, wyszukiwanie wyrazów pasujących do wzorca oraz podobnie brzmiących (opcjonalnie – również znajdowanie synonimów), dzięki wbudowanemu słownikowi MicroSpell firmy Trigram Systems lub Turbo-Lightning Borlanda;
- wspomaga dzielenie i przenoszenie wyrazów (ang. *hyphenation*) pochodzących z różnych języków;
- przechowuje wszystkie obiekty użytkowe (dokumenty, pliki sekwencji klawiszowych, klawiatury i in.) w zakodowanej formie na tzw. wolumenach; wolumen jest pojedynczym plikiem z punktu widzenia systemu operacyjnego;
- udostępnia operacje konwersji podstawowych elementów użytkowych, tzn. dokumentów i sekwencji klawiszowych, na czytelną postać zapisywaną do pliku tekstowego, który po ewentualnej modyfikacji może być poddany konwersji w odwrotną stronę;
- tekst zawierający znaki z rozszerzonego zestawu ASCII można wstawić do istniejącego dokumentu z uwzględnieniem przyporządkowania kodom 0–127 i 128–255 dwu różnych fontów T^3;
- (wersja 2.3 z 1990 r.) zawiera konwerter dokumentu T^3 na postać akceptowaną przez T_EX-a lub WordPerfecta;
- implementacja programu została wykonana dla licznego zestawu sterowników graficznych (Hercules, CGA do VGA, IBM 3270 z XGA, HP Vectra i in.) oraz drukarek (od 9-igłowych do laserowych i PostScriptowych); obejmuje ona dużą liczbę przygotowanych firmowo: fontów, sekwencji klawiszowych (ok. 1000, w tym 400 matematycznych o nazwach wzorowanych na T_EX-u) i klawiatur (m. in. matematyczna, chemiczna, Dvoraka, 3 klawiatury rosyjskie, 7 zachodnioeuropejskich);
- użytkownik może w naturalny sposób rozbudowywać program o nowe fonty, klawiatury, sekwencje klawiszowe i dokumenty wzorcowe;
- program jest dobrze przystosowany do wykorzystania przez wielu użytkowników o różnych wymaganiach (uprzywilejowany użytkownik główny, hasła, wolumeny, prawa dostępu do nich i do pozostałych obiektów, zbiory typy *profile* deniniujące wstępne wymagania użytkownika, kontekstowe podpowiedzi wyświetlane zawsze lub na żądanie, możliwość pracy w sieci);
- w koncepcji użytkowania programu T^3 dominuje podejście obiektowe.

Podstawy posługiwania się programem T^3

Twórcy programu T^3 wprowadzili własne nazewnictwo dla klawiszy funkcyjnych (⟨F1⟩ – ⟨F10⟩) oraz innych klawiszy z klawiatury PC, wykorzystywanych jako klawisze sterujące działaniem programu (rys. 1).

Najważniejszą rolę z punktu widzenia użytkownika odgrywają klawisze: ⟨Accept⟩ i ⟨Escape⟩, ⟨Menu⟩ – wyświetlający dostępną w danej chwili listę wariantów wyboru lub informacje o obiekcie, ⟨Msg⟩ – pozwalający wznowić pracę programu, wstrzymaną z określonego powodu (np. błędu użytkownika) albo skorzystać z krótkiej kontekstowej podpowiedzi (ang. help), ⟨HiLight⟩ – służący do wyróżniania obiektów lub tekstu, oraz ⟨Revise⟩ – uaktywniający wszelkiego rodzaju modyfikacje.

W koncepcji programu T^3 silnie uwidacznia się podejście obiektowe. Lista rodzajów (klas) obiektów występujących na poziomie głównym programu (ang. *main menu*) oraz specyficznych dla dokumentu jest zebrana na rys. 2. Obiekty (egzmeplarze) z głównego poziomu są przechowywane w wolumenach, a obiekty specyficzne dla dokumentu – w dokumencie. Nazwy poszczególnych obiektów (i innych elementów użytkowych) składają się z ciągu co najwyżej 16 znaków z rozszerzonego zestawu ASCII (0–255). Pełna specyfikacja obiektu z poziomu głównego (oprócz wolumenu) wymaga podania nazwy obiektu i nazwy wolumenu. Obiekty specyficzne dla dokumentu oraz wolumeny są w sposób jednoznaczny identyfikowane przez swoją nazwę.

Dla wszystkich klas obiektów z rys. 2 obowiązują dwie podstawowe zasady:

T³ (2.21) – podstawowe klawisze i funkcje sterujące

szary <+>: Accept potwierdzenie proponowanego wyboru w formularzu i menu, zakończenie redagowania dokumentu
<Esc>: Escape rezygnacja z proponowanego wyboru lub dalszego redagowania dokumentu

<F1>: HiLight <F2>: Goto <F3>: Search <F4>: Revise <F5>: Copy
<F6>: Undo <F7>: Marker <F8>: Face <F9>: Menu <F10>: Msg/help

szara <*>: Send
<5> na klawiaturze numerycznej: Rpt (*repeat*)

<Shift+Esc>: Panic button
<Shift+Rpt>: Cursor locate
<Shift+Menu> – menu pomocnicze (ang. *Anytime menu*)

<Ins> – punkt lub tryb rozpychania
<Del> – usunięcie znaku

strzałki < ← >, < ↑ >, < ↓ >, < → > – przesunięcie kursora o jeden znak lub wiersz
<strzałka+Rpt> – ciągły przesuw kursora
<Home>, <End> – przesunięcie kursora na początek/koniec wyświetlanego menu lub tekstu dokumentu
<Shift+Home>, <Shift+End> – przesunięcie kursora na początek/koniec menu lub tekstu dokumentu
<Shift + ↑ >, <Shift + ↓ > – przewijanie tekstu
<Shift + ← >, <Shift + → > – przesunięcie kursora na początek/koniec wiersza

<Alt+klawisz_alfanumeryczny> – wprowadzenie znaku z klawiatury uzupełniającej (np. <Alt+a>, <Alt+Shift+a>)
<Alt+HiLight> – przełączenie się na stałe na klawiaturę uzupełniającą

<Ctrl+Menu> – menu sekwencji klawiszowych dostępnych w aktualnie dołączonym pliku tych sekwencji
<Ctrl+Copy> – rozpoczęcie nagrywania (zapamiętywania) sekwencji klawiszowej
<Ctrl+Accept> – zakończenie nagrywania sekwencji
<Ctrl+*name*> – odtworzenie sekwencji klawiszowej o nazwie *name*

<LeftShift+PrintScreen>, <RightShift+PrintScreen> – graficzne kopie ekranu, różniące się wielkością i orientacją
<LeftShift+RightShift+PrintScreen> – wysuw papieru drukarki o jedną stronę (ang. *form–feed*)

a)

T³ (2.21) – dodatkowe klawisze i funkcje sterujące dostępne w czasie przetwarzania dokumentu

<PgUp>: 1/2 Up <PgDn>: 1/2 Dn

<Enter> – koniec wiersza tekstu <Shift+Enter> – początek nowego akapitu

<strzałka+symbol> – przesunięcie kursora do określonego znaku lub znacznika
<Del+sekwencja_klawiszy> – usunięcie porcji tekstu (także w formularzach)

<Shift+spacebar> – niepodzielna spacja (ang. *lockspace*)

szary < – >: Center – centrowanie tekstu wiersza między marginesami
<Center + → >, <Center + ← >, <Center + znak> – różne sposoby centrowania tekstu

<Tab> – wstawianie znacznika tabulacji (równoważne <Tab + ← >)
<Tab + → >, <Tab + Center>, <Tab + znak> – różne sposoby wyrównywania tekstu względem punktu tabulacji
<Shift+Tab> – chwilowy lewy margines (ang. *hanging indent*)

<Menu> – aktywizacja głównego menu dokumentu (ang. *Document Part menu*)

<ScrollLock>: Other – aktywizacja menu znaczników (ang. *Other Keys menu*)

<HiLight+sekwencja_klawiszy> – wyróżnienie porcji tekstu
<HiLight+Revise> – zmiana trybu wyróżniania wiersza (Full/Half)

<Goto+HiLight> – skok kursora do początku wyróżnionego tekstu
<Goto+Marker> – skok kursora do markera
<Alt+Goto+strzałka> – szybka zmiana klawiatury uzupełniającej i/lub standardowej

<Search+Accept> – ponowne wyszukiwanie/zastępowanie uprzednio zdef. wzorca

<Revise> – modyfikacja nazwy fontu, numeru i atrybutów znaku wskazywanego przez kursor lub modyfikacja nazwy obiektu związanego ze znacznikiem

<Undo> – anulowanie zmian wprowadzonych w dokumencie

<Marker> – wstawienie markera w położeniu kursora, na początku lub na końcu wyróżnionego tekstu

<Face> – zmiana atrybu (wytłuszczenie, podkreślenie, przekreślenie) znaku wskazywanego przez kursor
<Face+sekwencja_klawiszy> – zmiana atrybutu znaków, w chwilowo wyróżnionej porcji tekstu
<Face+Revise> – zmiana atrybutu wszystkich znaków wprowadzanych z klawiatur
<Shift+Face> – wyświetlenie rysunku klawiatury (–tur) z możliwością wprowadzenia wskazanego znaku

<Accept> – zakończenie redagowania dokumentu.
<Accept>+<Accept> – zapamiętanie zmian, kontynuacja edycji (*save without exit*)
<Esc> – rezygnacja z dalszego redagowania dokumentu
<Esc> + <Accept> – anulowanie zmian, zakończenie edycji (ang. *exit without save*)

b)

Rys. 1. T³ – klawisze i funkcje sterujące
a) podstawowe,
b) dodatkowe, dostępne podczas przetwarzania dokumentu

- najpierw należy wybrać obiekt, a dopiero potem rodzaj wykonywanej na nim operacji (obiekt ←metoda),
- nowe obiekty powstają jedynie (nie dotyczy to wolumenów) z już istniejących – przez skopiowanie i modyfikację kopii (dziedziczenie, zawieranie).

Pierwsza zasada oznacza, że najpierw musimy wybrać np. dokument, który chcemy przetwarzać, a dopiero potem określić, co zamierzamy z nim zrobić (modyfikować, wydrukować, usunąć itp.) Podejście to odbiega od typowego, w którym najpierw wybieramy operację, np. redagowanie dokumentu, a dopiero potem określamy jakiego dokumentu ma ona dotyczyć.

T³ (2.21)– rodzaje (klasy) obiektów
(na poziomie głównym i specyficzne dla dokumentu)

- Wolumen (ang. *volume*)
- Font (grupa fontów) (ang. *font group*)
- Klawiatura (ang. *keyboard*)
- Plik sekwencji klawiszowych (ang. *key sequence file*)
- Papier (ang. *paper*)
- Parametry drukarki (ang. *printer setup*)
- Dokument (ang. *document*)
  - format strony (ang. *page format*)
  - format wiersza (ang. *line format*)
  - tekst opatrzony nazwą (ang. *named text*)
  - parametry wydruku dokumentu (ang. *print data*)
  - obiekt graficzny (ang. *graphics object*)
- Użytkownik (ang. *user*)

Rys. 2. Wszystkie rodzaje (klasy) obiektów wykorzystywane przez T³

Oprócz operacji specyficznych dla obiektu każdego rodzaju, istnieje zestaw operacji standardowych wykonywalnych na (prawie) każdym z nich. Obejmuje on operacje: modyfikacji (*Revise*), kopiowania i modyfikacji kopii (*Copy and revise the copy*)), kopiowania (*Copy*), zmiany nazwy (*Rename*), usunięcia (*Delete*), wyróżniania obiektów w menu (⟨*HiLight*⟩), ustalenia praw dostępu (*Set access*), zmiany właściciela (*Change owner*) i wyświetlenia informacji o obiekcie (⟨*Menu*⟩). Trzy ostatnie z nich dotyczą tylko obiektów z poziomu głównego, natomiast kolejna operacja – wydruk (*Send/Print*) – stosuje się wyłącznie do wolumenów i dokumentów.

Standaryzacja wybranych operacji umożliwia przypisanie klawiszom sterującym funkcji niezależnych od kontekstu (m.in. ⟨HiLight⟩, ⟨Revise⟩, ⟨Copy⟩, ⟨Del⟩, ⟨Menu⟩, ⟨Send⟩, tzn. że użycie wskazanych klawiszy albo inicjuje, tam gdzie ma to sens, wykonanie operacji wynikającej z ich nazwy, albo jest przez program ignorowane.

Podstawowy sposób pracy z T³ polega na wypełnianiu formularzy (ang. forms) i (lub) wyborze obiektów oraz operacji z menu. Dla przykładu, rozpoczęcie modyfikacji dokumentu wymaga wypełnienia dwu pól formularza o nazwie SELECT DOCUMENT określających nazwę dokumentu i nazwę wolumenu, zaakceptowania postaci tego formularza (⟨Accpet⟩) oraz wybrania i zaakceptowania operacji *Revise* z pojawiającego się automatycznie menu OPERATIONS. Użycie klawisza ⟨Menu⟩ w trakcie wypełniania formularza umożliwia wybranie nazwy wolumenu i (lub) nazwy dokumentu z odpowiednich menu. Modyfikację dokumentu można rozpocząć bez przechodzenia do menu OPERATIONS – wystarcza posłużyć się klawiszem ⟨Revise⟩ w chwili, gdy są wypełnione już oba pola formularza SELECT DOCUMENT.

Mechanizm skróconego wyboru operacji działa równie skutecznie dla wszystkich dopuszczalnych w T³ par *obiekt-operacja* takich, w których *operacja* ma przypisany sobie klawisz sterujący.

Zasada pierwsza – najpierw *obiekt*, potem *operacja* – obowiązuje również dla elementów nie wyszczególnionych na rys. 2, np. stosuje się do operacji na fragmentach tekstu.

Odzwierciedleniem zasady drugiej, tzn. tworzenia nowych obiektów na podstawie starych, jest występowanie operacji *Copy and revise the copy* w (prawie) każdym menu operacji na obiektach.

Naturalną konsekwencją wynikającą z tej zasady w odniesieniu do dokumentów jest przygotowanie zestawu dokumentów wzorcowych (ang. *shell documents*), stanowiących swoistą bibliotekę gotowych tekstów, formatów stron i wierszy, nagłówków, parametrów wydruku itp. Staranne opracowanie takiej biblioteki wzorców pozwala znacznie uprościć redagowanie i formatowanie dokumentów określonego typu (artykułów, raportów lub książek), które powinny trzymać się pewnych narzuconych standardów wydawniczych.

T³ nie udostępnia mechanizmu tworzenia obiektów innego rodzaju niż wyszczególnione w spisie rys. 2. Z drugiej strony, T³ niej pozwala na usunięcie z dokumentu wszystkich (specyficznych dla dokumentu) obiektów określonego rodzaju, np. wszystkich obiektów graficznych lub wszystkich formatów stron. W każdym dokumencie istnieje więc zawsze co najmniej jeden taki obiekt, co decyduje o samowystarczalności dokumentu w odniesieniu do zasady drugiej.

Dominujące w T³ podejście obiektowe ułatwia przyszłą rozbudowę programu, przy jednoczesnym zachowaniu ujednoliconych zasad korzystania z niego.

Elementy użytkowe programu T³ i ich wykorzystanie

Wolumeny

Wolumeny mogą przechowywać wszystkie rodzaje obiektów wykorzystywanej przez T³, oprócz obiektów własnego typu. Stanowią one zamkniętą całość, mając własną strukturę wewnętrzną niedostępną dla użytkownika. W razie potrzeby rozmiar wolumenu zwiększa się automatycznie, aż do osiągnięcia maksymalnej wielkości 2 MB.

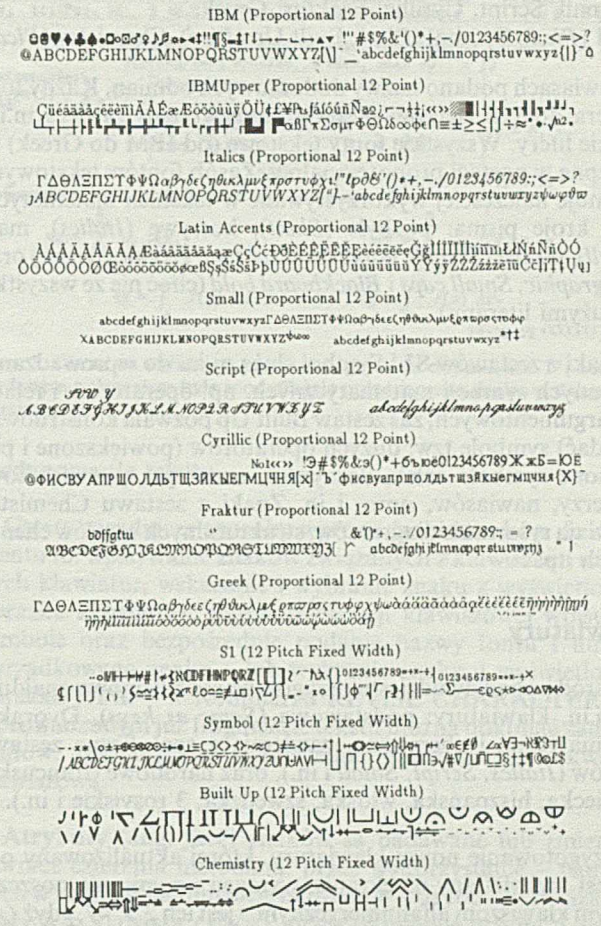
Operacje specyficzne dla wolumenu obejmują m.in. kopiowanie zawartości (wybiórcze lub całościowe), kontrolę wewnętrznej struktury (*Check*), odbudowę tej struktury w razie wykrycia błędu (*Recover*) i usunięcie całej zawartości (*Initialize*).

Wolumeny są, z kilku powodów, obiektem nietypowym w zestawieniu z rys. 2. Tworzy się je od zera (*Make a new volume*), nie jest przewidziana dla nich operacja modyfikacji, zaś dostępność wolumenu z programu T³ wymaga jego uprzedniego zamontowania (*Mount*).

Wolumeny są jednym z podstawowych elementów koncepcji programu T³, ułatwiającym: zachowanie podziału tematycznego i porządku w pracy z komputerem, realizację indywidualnych potrzeb czy wymianę informacji. Wolumen zawierający np. specyficzne dla pewnej tematyki i klawiatury, sekwencje klawiszowe i dokumenty stanowi logiczną całość, w niewielkim stopniu zależną od otoczenia i dającą się łatwo manipulować

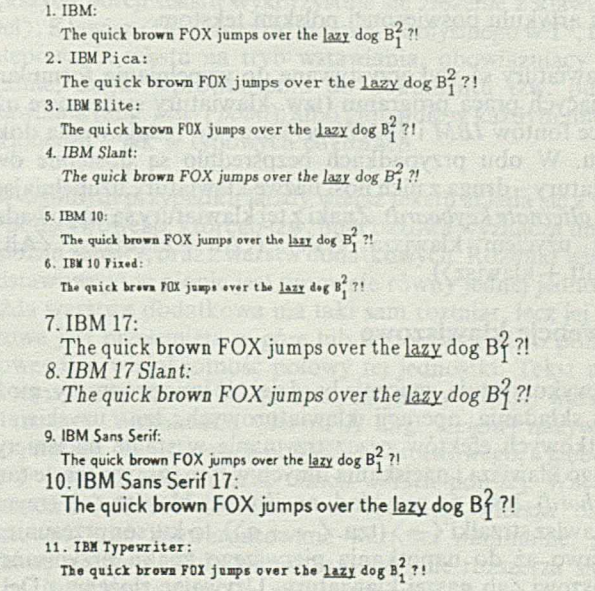
Fonty

Firma TCI dostarcza 13 podstawowych zestawów znaków (rys. 3) dostępnych w 40 odmianach (fontach) różniących się krojem i wielkością od wersji podstawowej. Są to:



Rys. 3. Komplet zestawu znaków używanych przez T³

- IBM (*Pica*, 10, 10 *Fixed*, 17, 17 *Slant*, *SansSerif*, *SansSerif* 17, *Elite*, *Slant*, *Typewriter*) – (rys. 4),



Rys. 4. Próbki fonów IBM

- IBMUpper (*Pica*, 17, 17 *Slant*, *Elite*, *Slant*),
- Italics (*Pica*, 10, *Elite*),

- Latin Accents (*Pica*, 17, 17 *Slant*, *Elite*, *Slant*),
- Small, Script, Cyrillic, Fraktur, Greek,
- S1 (*Pica*), Symbol (*Pica*), Built Up (*Pica*), Chemistry (*Pica*).

W nawiasach podano nazwy dodatkowych odmian. Każdy font zawiera do 128 znaków. Zestaw *Latin Accents* zawiera m.in. polskie litery. Wszystkie fonty tekstowe (od IBM do Greek) są dostępne w postaci proporcjonalnej. Zasób fontów tekstowych obejmuje najczęściej wykorzystywane w tekstach matematycznych kroje pisma: pochyłe (*Slant*), kursywę (*Italics*), małe (*Small*), pisane (*Script*), gotyckie (*Fraktur*), greckie (*Greek*) oraz *Calligraphic*, *Small caps* i *Blackboard bold* (choć nie ze wszystkimi dużymi literami).

Znaki z zestawów S1 i Symbol służą m.in. do wprowadzania wybranych symboli matematycznych, np. operatorów i relacji dwuargumentowych, zaś zestaw Built Up pozwala konstruować (składać) symbole tzw. dużych operatorów (powiększone i pogrubione symbole całek, iloczynów, sum itp.), pierwiastków, macierzy, nawiasów, norm i in. Znaki z zestawu Chemistry ułatwiają rysowanie diagramów, strukturalnych wzorów chemicznych itp.

Klawiatury

Wśród ponad dwudziestu przygotowanych firmowo znajdują się m.in. klawiatury: podstawowa (*Same as keys*), Dvoraka, matematyczna i chemiczna, zawierające specyficzne zestawy znaków (*Italics*, *Script*, *Small* i in.), oraz narodowe (francuska, niemiecka, hiszpańska, włoska, szwedzka, 3 rosyjskie i in.).

Przygotowanie nowej klawiatury, której aktualizowany obraz jest wyświetlany na ekranie, wymaga przypisania poszczególnym klawiszom alfanumerycznym – jest ich 2 × 47, gdyż ⟨a⟩ i ⟨A⟩ są traktowane jako oddzielne klawisze alfanumeryczne – znaków należących do istniejących (być może różnych) fontów. Dodatkowo można określić, które klawisze mają reagować na ustawienie górnego rejestru klawiatury (⟨CapsLock⟩). Ponadto można zwiększyć liczbę symboli wprowadzanych z użyciem tej klawiatury (np. o specyficznie akcentowane litery) posługując się zdefiniowanymi przez siebie klawiszami nieaktywnymi, (ang. *dead keys*) – tak jak opisano to w dalszej części artykułu poświęconej polskim tekstom.

Klawiatury są wykorzystywane do wypełniania formularzy sterujących pracą programu (tzw. klawiatury systemowe używające fontów *IBM* i *IBMUpper*) oraz do redagowania dokumentu. W obu przypadkach bezpśrednio są dostępne dwie klawiatury – druga z nich nosi nazwę klawiatury uzupełniającej (ang. *alternate keyboard*). Znaki z tej klawiatury są wprowadzane z użyciem klawisza ⟨Alt⟩ (⟨Alt + klawisz⟩, ⟨Alt + + Shift + klawisz⟩).

Sekwencje klawiszowe

T³ wykorzystuje, znacznie bardziej niż inne programy, możliwość składania operacji klawiaturowych, tzn. uzyskiwania dodatkowych efektów przez trzymanie w stanie naciśniętym jednego klawisza i naciskania innych w tym samym czasie (ang. *key-chord*). Jeśli, na przykład, naciśniemy klawisz ⟨a⟩ trzymając klawisz strzałki ⟨→⟩ (tzn. ⟨→ + a⟩), to kursor przesunie się w prawo aż do napotkania pierwszego znaku przypisanego klawiszowi ⟨a⟩ naszej klawiatury. Używając złożenia ⟨Del + + Enter ←⟩ usuniemy tekst od aktualnego położenia kursora do najbliższego znacznika końca wiersza, ale bez niego. Jeśli jednak przed zwolnieniem klawisza ⟨Del⟩, użyjemy jeszcze ⟨Esc⟩, to po prostu zrezygnujemy z operacji usunięcia tej porcji tekstu.

Uzyskanie powyższych efektów jest możliwe dzięki temu, że T³ rozróżnia stan naciśnięcia i zwolnienia klawisza, a pełne wykonanie operacji związanej z danym klawiszem sterującym (np. ⟨Del⟩, ⟨Revise⟩, ⟨Menu⟩) rozpoczyna się dopiero po jego zwolnieniu.

Kolejną bardzo ważną właściwością T³ jest możliwość nadania nazwy i zapamiętania ciągu operacji klawiaturowych (sekwencji klawiszowej) w tzw. plikach sekwencji klawiszowych (ang. *key sequence file*). Odtwarzanie zapisanej sekwencji o nazwie *name* odbywa się w bardzo naturalny sposób – przez ⟨Ctrl + name⟩ (tj. naciśnięcie ⟨Ctrl⟩, wpisanie nazwy *name*, zwolnienie ⟨Ctrl⟩) lub wybór z menu. T³ nie wprowadza ograniczenia na liczbę sekwencji klawiszowych ani na zakres wykonywalnych przez nie operacji.

Sekwencje klawiszowe mogą odwoływać się do innych sekwencji, a nawet do samych siebie (⟨PanicButton⟩ przerywa wykonanie takiej nieskończonej pętli), ale nie mogą zawierać parametrów ani konwersować z użytkownikiem. Operacje przewidziane dla sekwencji klawiszowych to m.in.: wypisanie w czytelnej postaci do pliku tekstowego, zmodyfikowanie z wykorzystaniem dodatkowych konstrukcji oraz ponowne wczytanie do pliku sekwencji klawiszowych (ASCII Export/Import).

Sekwencja klawiszowa jest przechowywana jako ciąg liczb odpowiadających kodom naciśniętych i zwolnionych klawiszy. Nie są zapamiętywane aktualne znaki przypisane klawiszom alfanumerycznym klawiatur używanych w chwili zapisywania. Dlatego zapisanie wzoru wprowadzonego np. z użyciem klawiatury matematycznej (Math) i jego późniejsze odtworzenie przy zmienionej klawiaturze spowoduje wygenerowanie wzoru składającego się z innych symboli.

Bardzo istotna staje się zatem możliwość przygotowania sekwencji klawiszowej (wprowadzającej symbole do dokumentu) w taki sposób, że jej efekt jest niezależny od aktualnie przyłączonych klawiatur, a jedynie – co oczywiste – od dostępności fontów. Tego typu sekwencje klawiszowe (rys. 5) zawsze wykorzystują operację modyfikacji pojedynczego znaku w dokumencie (ang. *revise character*).

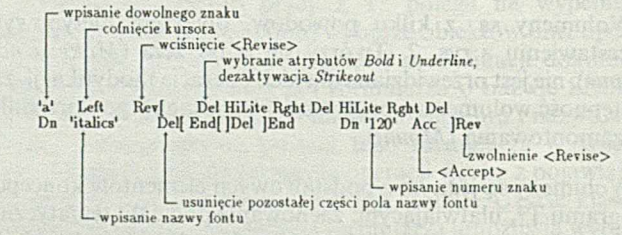
T³ (2.21) – przykład sekwencji klawiszowej
działającej niezależnie od przyłączonych klawiatur

1. Sekwencja klawiszowa *itx* wpisuje literę 'x' z fontu 'Italics', mającą atrybuty: wytłuszczenie i podkreślenie (ang. *Bold*, *Underline*):

   Ξ Ξ Ξ...

2. Sekwencja *itx* zdekodowana na postać ASCII:

START OF KEY SEQUENCE itx

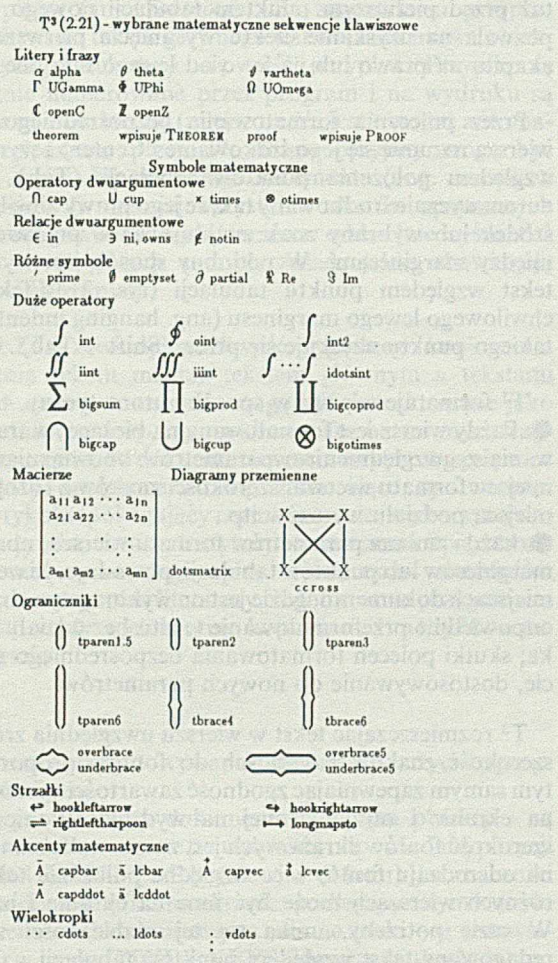


Rys. 5. Przykładowa sekwencja klawiszowa T³, zapisana w czytelnej formie

T³ umożliwia zapisywanie sekwencji klawiszowych w taki sposób, że podczas ich odtwarzania nie są wyświetlane na ekranie formularze i menu, z których korzystano. W efekcie końcowym sprawia to wrażenie, że program został rozbudowany o nowe polecenia dostępne przez klawisz ⟨Ctrl⟩.

Sekwencje klawiszowe przygotowane firmowo są zapisane tak, aby działały niezależnie od przyłączonych klawiatur oraz

minimalizowały informacje wyświetlane na ekranie. Tematycznie można je podzielić na matematyczne, chemiczne i pomocnicze (dołączanie klawiatur, zamiana jednego fontu na drugi, zmiana atrybutów znaku itp.). Wybrane sekwencje matemtyczne opisano na rys. 6.



Rys. 6. Kilka przygotowanych firmowo, matematycznych sekwencji klawiszowych T³

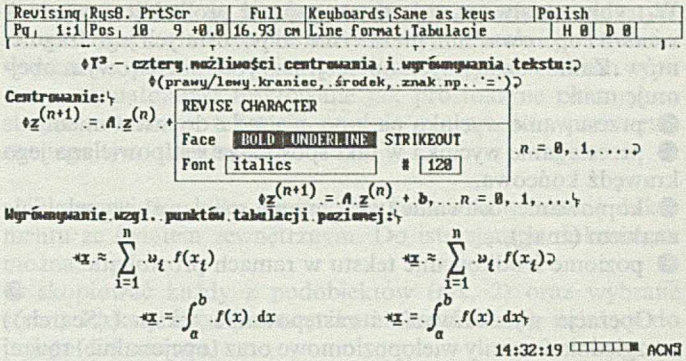
Dokumenty

Postać ekranu

Na górze ekranu (rys. 7) są wyświetlane (opcjonalnie) trzy wiersze informacyjne, podające: nazwę dokumentu, tryb wyróżniania wiersza (*Full/Half*), nazwy dwu bezpośrednio dostępnych klawiatur, numer strony i regionu, pozycję w wierszu, numer wiersza i położenie kursora względnie poziomu bazowego (o ile poniżej lub powyżej), odległość (w cm, calach lub punktach) kursora do dolnej krawędzi regionu, nazwę formatu wiersza obowiązującego w miejscu, gdzie znajduje się kursor wraz z jego wysokością i głębokością (*H, D*), oraz graficzny wzorzec tegoż formatu z zaznaczonymi marginesami, punktami tabulacji i położeniem kursora.

W dolnym prawym rogu są wyświetlane (zawsze): aktualna godzina, stan wybranych trybów pracy klawiatury (*A* – klawiatura uzupełniająca, *C* – CapsLock, *N* – NumLock, kluczyk – zapisywanie sekwencji klawiszowej) oraz 8 prostokątnych „lampek" sygnalizujących działanie jednostki centralnej, klawiatury i in.

Dla tekstu właściwego przeznaczono 17 albo 31 pojedynczych wierszy, w zależności od trybu wyświetlania – normalnego albo zagęszczonego.



Rys. 7. Obraz na ekranie w czasie redagowania dokumentu T³; ostatnim wydanym poleceniem była modyfikacja pojedynczego znaku w dziewiątym wierszu

Redagowanie tekstu

Cztery zasadnicze sposoby wprowadzania znaków do dokumentu to: wpisywanie znaków związanych z klawiszami używanych klawiatur, wskazanie i wybranie znaku z wyświetlonego obrazka klawiatury, użycie sekwencji klawiszowej wpisującej symbole oraz bezpośrednie podanie nazwy fontu i numeru porządkowego znaku – lub wybranie znaku z wyświetlonego obrazka fontu – w formularzu REVISE CHARACTER. We wprowadzonym już fragmencie tekstu można dokonać zamiany fontów, posługując się odpowiednią operacją lub sekwencją klawiszową.

Atrybuty (ang. *face*) znaków są nadawane lub zmieniane również czterema metodami: przez wykorzystanie – pamiętanych w każdym dokumencie oddzielnie – atrybutów związanych z użyciem klawisza ⟨Face⟩, przez określenie atrybutu wszystkich znaków wprowadzanych z użyciem klawiatur (⟨Face + + Revise⟩), przez użycie sekwencji klawiszowej (lub operacji) zmiany atrybutów fragmentu tekstu oraz przez zmianę atrybutu znaku w formularzu jego modyfikacji.

Ciągłe przesuwanie kursora uzyskuje się składając naciśnięcie klawisza strzałki z naciśnięciem klawisza ⟨Rpt⟩, a do usunięcia większych porcji tekstu wykorzystuje się złożenie z klawiszem ⟨Del⟩. Klawisz ⟨Ins⟩ służy do zmiany, domyślnego w T³, trybu zastępowania tekstu na tryb wstawiania, obowiązujący albo lokalnie, dzięki wprowadzeniu do dokumentu tzw. punktu wstawiania (ang. *insert point*), albo globalnie w każdym miejscu dokumentu – jak w typowych edytorach.

W ogólnym przypadku każdy wiersz tekstu składa się z wiersza podstawowego, którego tzw. linia bazowa wyznacza umowny pozim zerowy, oraz z warstw dodatkowych. Rozmiar wiersza podstawowego w pionie jest umownie równy jednej jednostce. Każda warstwa dodatkowa ma taki sam rozmiar, lecz jej linia bazowa jest przesunięta w górę lub w dół (względem pozomu zerowego) o wielokrotność połowy tej jednostki. Tekst może być wpisywany w wierszu podstawowym i w każdej warstwie dodatkowej. Wysokością wiersza *H* ang. *Height*) nazywamy liczbę warstw dodatkowych powyżej podstawowego, a głębokością *D* (ang. *Depth*) – liczbę takich warstw poniżej. Wartości parametrów *H* i *D* określane w formacie wiersza należy traktować jako zagwarantowane wartości minimalne, gdyż w miarę potrzeb każdy wiersz samoczynnie rozszerza się ku górze i (lub) ku dołowi, aby zrobić miejsce na kolejne poziomy symboli. Do przesuwania kursora w pionie, w ramach jednego wiersza, służą klawisze ⟨1/2 Up⟩ i ⟨1/2 Dn⟩.

Sposób traktowania wiersza w pionie przez T³ zależy od bieżącej wartości trybu wyróżniania (ang. *highlight*) wiersza, którym może być tryb całościowy (Full) lub warstwowy (Half).

W trybie warstwowym można wyróżnić prostokątny wycinek wiersza i operować nim niezależnie od pozostałych jego fragmentów. Zakres dostępnych operacji w trybie warstwowym obejmuje m.in.:

- przesuwanie wycinka na inne miejsce z dopasowaniem,
- przeciąganie wycinka w taki sposób, że jest powielana jego krawędź końcowa,
- kopiowanie, usuwanie (wypełnianie spacjami), wypełnianie znakiem (znak),
- poziome środkowanie tekstu w ramach prostokąta.

Operacja wyszukiwania i zastępowania tekstu (⟨Search⟩) uwzględnia formuły wielopoziomowe oraz (opcjonalnie) rodzaj fontu, atrybuty, rozróżnienie małych i dużych liter. We wzorcach wyszukiwania i zastępowania nie przewidziano jednak możliwości użycia metasymboli zaczerpniętych z konstrukcji wyrażeń regularnych.

Fragmenty tekstu

W T^3 istnieje zestaw operacji przewidzianych do stosowania na fragmentach tekstu. Przez fragment tekstu rozumie się tutaj: tekst wyróżniony w trybie całościowym (*Full*), strony oraz tekst między znacznikami (markerami). Znaczniki (niewidoczne na ekranie) służą do przypisania nazw położeniom ważnych fragmentów dokumentu, np. początkowi nowego rozdziału lub istotnemu wzorowi.

Użycie klawisza ⟨Goto⟩ pozwala szybko przesuwać kursor do początku strony o określonym numerze lub o określną liczbę stron, na początek wyróżnionego tekstu czy wreszcie do znacznika o podanej nazwie. Wyróżnienie tekstu nie wymusza natychmiastowego wykonania na nim jakiejś operacji, jak w niektórych procesorach tekstu działających wg zasady: *cut and paste*.

Dalsze operacje dostępne na fragmentach tekstu lub w ich zakresie to m.in.:

- kopiowanie, przesuwanie i usuwanie,
- kopiowanie i przesuwanie do tekstu opatrzonego nazwą,
- wyszukiwanie i zastępowanie,
- dzielenie i przenoszenie wyrazów (ang. *hyphenate*),
- kontrola pisowni (ang. *spell check*), znajdowanie wyrazów zbliżonych wymową, pisownią lub znaczeniem.

T^3 nie dzieli wyrazów w sposób automatyczny, tzn. korzystając ze stosownego algorytmu lub słownika. Zamiast tego użytkownik ma możliwość wprowadzenia łączników (ang. *hyphens*) do wpisywanego tekstu, uwzględnianych jako potencjalne miejsca przenoszenia wyrazów podczas automatycznego formatowania wierszy lub wykonania operacji dzielenia i przenoszenia wyrazów w trybie konwersacji z programem.

Formatowanie

Zasadnicze formatowanie dokumentu obejmuje: jego podział na strony i rozplanowanie położenia tekstu na stronie na podstawie formatów stron, podział na wiersze w ramach strony na podstawie formatów wierszy oraz rozmieszczenie tekstu w ramach wiersza z ewntualnym wykorzystaniem dodatkowych poleceń formatowania bezpośredniego. Sposoby formatowania znaków (fonty, atrybuty) omówiono w punkcie poświęconym redagowaniu tekstu.

Format wiersza określa m.in. wysokość H i głębokość D wspomniane uprzednio, ew. sposób wyrównania tekstu do prawego marginesu (przez mikrojustację) oraz rozmieszczenie marginesów i punktów tabulacji poziomej, mierzone w bezwzględnych jednostkach długości. Zmianę obowiązującego formatu wiersza uzyskuje się przez wprowadzenie do tekstu dokumentu znacznika formatu wiersza (ang. *line format reference*) i związane z nim nazwy przygotowanego wcześniej formatu.

Znacznik początku akapitu ⟨Shift + Enter⟩ jest wstawiany tuż przed pierwszym punktem tabulacji nowego wiersza, co pozwala na uzyskanie efektu wysunięcia pierwszego wiersza akapitu na prawo lub na lewo od lewego marginesu.

Przez polecenia formatowania bezpośredniego w ramach wiersza rozumie się jego śrdkowanie ⟨Center⟩ i wyrównywanie względem położenia punktów tabulacji ⟨Tab⟩. Tekst jest automatycznie środkowany tak, że jego prawy albo lewy koniec, środek lub wybrany znak znajduje się w połowie odległości między marginesami. W podobny sposób jest wyrównywany tekst względem punktu tabulacji (rys. 7). Efekt ustalenia chwilowego lewego marginesu (ang. hanging indent) względem takiego punktu uzyskuje się przez ⟨Shift + Tab⟩.

T^3 formatuje wiersze w sposób automatyczny, tzn.:

- każdy wiersz jest formatowany na bieżąco, w trakcie wpisywania z uwzględnieniem parametrów obowiązującego w tym miejscu formatu wiersza, szerokości znaków z różnych fontów, miejsca podziału wyrazów itp.,
- każda zmiana parametrów formatu wiersza, np. wysokości, marginesów lub punktów tabulacji, powoduje, że we wszystkich miejscach dokumentu, gdzie jest on wykorzystywany, następuje odpowiednie przeformatowanie tekstu bez udziału użytkownika; skutki poleceń formatowania bezpośredniego są, oczywiście, dostosowywanie do nowych parametrów.

T^3 rozmieszczając tekst w wierszu uwzględnia zróżnicowaną szerokość znaków należących do fontów proporcjonalnych, tym samym zapewniając zgodność zawartości wiersza widzianej na ekranie i otrzymywanej na wydruku. Ponieważ jednak szerokość fontów ekranowych jest zawsze taka sama i niezależna od rodzaju fontu, więc względne położenie tekstu w dwu różnych wierszach może być inne na ekranie i na wydruku. W razie potrzeby, unika się tej rozbieżności wyrównując redagowany tekst względem punktów tabulacji.

Początki stron dokumentu są wyznaczane przez położenie odpowiednich znaczników (ang. *page start*) wstawianych do jego tekstu:

- „ręcznie", w czasie redagowania dokumentu przez użytkownika,
- podczas podziału dokumentu na strony z użyciem operacji *Paginate* (dla wybranych stron lub całego dokumentu) dopuszczającej interwencję użytkownika,
- automatycznie przez T^3, przed wykonaniem wydruku.

Z każdym znacznikiem początku strony są związane nazwy dwu formatów stron. Jeden z nich (ang. *primary*) obowiązuje dla najbliższej strony, a drugi (ang. *secondary*) dla następnej po niej i pozostałych. T^3 ułatwia w ten sposób przygotowanie dokumentów wzorcowych, stosujących się do schematu formatowania przyjmowanego zwykle dla artykułów w czasopismach i materiałach konferencyjnych lub dla rozdziałów większych opracowań.

Format strony określa rozmieszczenie na stronie od 1 do 8 tzw. reginów, czyli prostokątnych obszarów wypełnionych tekstem głównym dokumentu (ang. *main text*) lub wcześniej przygotowanymi tekstami opatrzonymi nazwą (ang. *named text*). Parametry regionu określają jego numer oraz mierzone w jednostkach bezwzględnych położenie i rozmiary. Region wypełniony tekstem opatrzonym nazwą (lub dwoma – innym dla strony parzystych i nieparzystych) spełnia najczęściej rolę nagłówka lub stopki (ang. *header, footer*). Naturalnym wykorzystaniem kilku regionów z tekstem głównym jest rozmieszcze-

nie tekstu dokumentu w wielu łamach na stronie (ang. *multiple columns*). Łamy te mogą się oczywiście różnić szerokością i wysokością.

Teksty opatrzone nazwą są również wykorzystywane jako teksty przypisów (ang. *footnotes*). Ze znacznikiem przypisu wprowadzonym do tekstu głównego jest wiązana nazwa obiektu, w którym przechowuje się treść przypisu. Przypisy są automatycznie numerowane przez program i na wydruku są umieszczane na dole strony (miejse na nie jest również automatycznie uwzględniane przez T^3; długie przypisy mogą być dzielone na części) lub na końcu dokumentu (ang. *endnotes*). Treść przypisów (i ogólniej wszystkich tekstów opatrzonych nazwą) jest redagowana i formatowana w oddzielnie wyświetlanym oknie, według tych samych zasad, jakie obowiązują w dokumencie głównym.

Ponieważ T^3 przewiduje możliwość wygodnego kopiowania i przenoszenia tekstu między tekstem głównym a tekstami opatrzonymi nazwą, te ostatnie mogą być potraktowane jako swoista składnica typowych, prawidłowo sformatowanych fragmentów dokumentu. W dokumencie wzorcowym mogą się więc pojawić teksty opatrzone nazwą, zawierające np. wzorzec początku artykułu (pokazujący rozmieszczenie i wykorzystywane fonty dla tytułu, autora, instytucji, streszczenia itp.), strony tytułowej podręcznika, jego spisu treści, nazw rozdziałów i podrozdziałów czy spisu literatury. Jednokrotnie przygotowane wzorce są wtedy wielokrotnie kopiowane i modyfikowane aktualną treścią.

Wiązania

Ogólne zasady przygotowywania dokumentu z użyciem programu T^3 mówią, że:

- parametry i elementy dokumentu – w tym obiekty – są modyfikowane lub tworzone po wybraniu odpowiedniej pozycji z menu DOCUMENT PARK (klawisz ⟨Menu⟩), a znaczniki – symbole widoczne tylko na ekranie, podlegające tym samym regułom przenoszenia lub usuwania co inne znaki – są wstawiane do tekstu w menu OTHER KEYS (klawisz ⟨Other⟩) lub po użyciu klawiszy sterujących ⟨Enter⟩, ⟨Center⟩ i ⟨Tab⟩,
- z wybranymi typami znaczników związane są nazwy odpowiednich obiektów przypisywane im domyślnie przez T^3 lub zmieniane przez użytkownika w wyniku operacji *Revise character* (klawisz ⟨Revise⟩), jest to tzw. wiązanie.

Zasada wiązania obejmuje znaczniki: formatu wiersza, początku strony, przypisu, obiektów graficznych oraz tzw. symboli wypełnianych podczas wydruku (ang. *print time fill-in*). Wśród pozycji nie mających nic wspólnego ze znacznikami warto wymienić m.in. parametry charakteryzujące dokument, takie jak: tablice używanych fontów i klawiatur (w tym dwu bezpośrednio dostępnych), atrybuty znaków przypisane klawiszowi ⟨Face⟩, wartości i atrybuty charakteryzujące liczniki stron i przypisów, nazwa pomocniczego słownika.

Obiekty graficzne, komunikacja ze światem zewnętrznym

T^3 uwzględnia możliwość wkomponowania w tekst dokumentu obrazów graficznych przygotowanych przez inne programy (ang. graphics import). Parametry obiektu graficznego określają: nazwę pliku zawierającego obraz graficzny, jego oryginalną i oczekiwaną na wydruku wysokość oraz szerokość (skalowanie) lub sposób dobierania proporcji, jak również definiują, że tekst właściwy dokumentu ma otaczać grafikę z lewej lub prawej strony.

Znaczniki obiektów graficznych wstawiane do dokumnetu są typu *Large* albo *Small*. Obiekty graficzne związane ze znacznikiem typu *Small* są traktowane podczas formatowania jak element wiersza tekstu o zdefiniowanym wcześniej rozmiarze w pionie i poziomie; obiekty związane ze znacznikiem typu *Large* są natomiast traktowane jak prostokątne obszary na stronie, otoczone tekstem, zgodnie z wartością parametru znanego tym obiektom.

T^3 przewiduje kilka możliwości komunikowania się dokumentu ze światem zewnętrznym. Do istniejącego dokumentu można:

- skopiować każdy z podobiektów (rys. 2) oraz wybrane strony i teksty między znacznikami (markerami), należące do innego dokumentu,
- wstawić tekstowy plik ASCII w miejsce wskazywane przez kursor.

Dodatkowe możliwości konwersji dokumentu na postać pliku ASCII daje, omówiona dalej, operacja *DOS Transfer*.

Operacje specyficzne dla dokumentu

Naciśnięcie ⟨Acept⟩ lub ⟨Esc⟩ na głównym poziomie redagowania dokumenu uaktywnia trójelementowy formularz pozwalający użytkownikowi zdecydować się na zapamiętanie zmian i zakończenie pracy nad dokumentem z ewentualnym wykonaniem wydruku. Przez zmiany w dokumencie rozumie się tutaj zmiany jakiegokolwiek elementu składowego dokumentu, tj. tekstu, formatu strony, parametrów obiektu graficznego itd.

Specyficzne operacje dla dokumentu to m.in.:

- wydruk (*Send/Print*) i wydruk łączony (*Merge print*),
- konwersja na postać ASCII (*DOS Transfer*).

Wydruk nie jest możliwy w trakcie redagowania dokumentu. Zakres wydruku został ograniczony do całego dokumentu lub wybranych stron. T^3 pozwala na łączenie – podczas wydruku – dokumentów zawierających zmienne pola (ang. *merge fields*), instrukcje przypisania i warunkowe, z danymi podawanymi z klawiatury, pliku tekstowego ASCII lub dokumentu T^3. Użycie dokumentu T^3 jako danych umożliwia drukowanie tekstów łączonych opartych na całym zestawie znaków dostępnym w T^3, a więc tekstów wielojęzycznych, wzorów matematycznych, chemicznych itd.

Obie operacje wydruku dokumentu odwołują się w swoich formularzach m.in. do nazw dwu obiektów sterujących. Jednym z nich jest obiekt typu *Print data*, który określa parametry wydruku narzucone przez dokumet i niezależne od rodzaju drukarki, drugim natomiast – obiekt typu *Printer setup*, którego najważniejszym parametrem jest nazwa pliku tekstowego sterującego pracą konkretnej drukarki (np. Epson LQ 1500). Zawartość tego pliku została starannie opisana w dokumentacji technicznej T^3.

Wykonanie operacji *DOS Transfer* dla dokumentu T^3 spowoduje utworzenie pliku tekstowego ASCII zawierającego – zapisane w określonej konwencji – wszystkie lub wybrane informacje składające się na dokument, tj. tekst główny, teksty opatrzone nazwą i inne podobiekty dokumentu (rys. 2) oraz jego pozostałe parametry. Po ewntualnej modyfikacji otrzymanego pliku lub przesłaniu go pocztą elektroniczną, T^3 umożliwia utworzenie na jego podstawie nowego dokumentu w istniejącym już wolumenie (*Import a document*). W nowszej wersji programu T^3 [8] wykorzystano rezultat operacji *DOS Transfer* jako etap pośredni konwersji dokumentu na postać akceptowaną przez T_EX-a lub WordPerfecta.

Do najważniejszych – z punktu widzenia przygotowywania dokumentów – braków programu T^3, traktowanego jako wysokiej klasy program TWP, wypada zaliczyć:

● brak narzędzia do wygodnego rysowania linii i prostokątnych ramek;
● brak możliwości: automatycznego generowania wielopoziomowego skorowidza i spisów zawartości (spis treści, zestawienia rysunków i tabel itp.), automatyzacji posługiwania się odwołaniami do bibliografii i obsługi formowania jej spisu, automatycznej numeracji (wzorów, twierdzeń, tabel itp.), odsyłaczy symbolicznych w przód i w tył do numerowanych automatycznie pozycji – co staje się szczególnie istotne podczas opracowywania większych dokumentów;
● brak opcji zmiany kolejności materiału z kolejności wejściowej na poprawiającą wygląd dokumentu (ang. *floating text*), np. przemieszczenia na następną stronę tabeli nie mieszczącej się w całości na stronie bieżącej i wypełnienia tekstem leżącym poniżej tej tabeli powstałych w ten sposób pustych obszarów na stronie.

Z formalnego punktu widzenia, T³ nie w pełni spełnia wymagania definicji programu z interfejsem WYSIWYG, choć jest zawsze do tej grupy zaliczany. Najważniejsze odstępstwa to:
● fonty proporcjonalne są wyświetlane na ekranie w postaci fontów o stałej szerokości,
● znaki drukowane na drukarce odbiegają wyglądem i jakością od swoich odpowiedników na ekranie,
● znaczniki (koniec wiersza, symbol tabulacji, początek strony itp.) są zwykle wyświetlane na ekranie, zawsze natomiast pomijane na wydruku,
● akapity są wyrównywane do prawego marginesu (justacja) dopiero na wydruku,
● regiony są rozmieszczane na ekranie (w kolejności numerów) jeden pod drugim niezależnie od ich rzeczywistego rozmieszczenia na stronie wydruku – w kolejnej wersji programu ma być poprawiona ta niedogodność,
● teksty opatrzone nazwą (nagłówki, przypisy) nie są wyświetlane na ekranie zgodnie z ich rozmieszczeniem na drukowanej stronie, choć informacje o nich są zapisane w formatach stron lub w znacznikach przypisów.

Użytkownicy

T³ jest programem dobrze przystosowanym do tego, aby korzystało z niego wielu użytkowników o różnych wymaganiach. Każdy użytkownik ma swój identyfikator i hasło oraz zbiór typu *profile* definiujący jego parametry. Obejmują one m.in.:
● domyślnie wybierany dokument, plik sekwencji klawiszowych, obiekt typu *Printer Setup* i poziom kontekstowych podpowiedzi (*help*),
● dwie używane klawiatury systemowe,
● dodatkowe ścieżki (ang. *paths*) przeszukiwane w celu automatycznego montowania wolumenów,
● opcje redagowania dokumentów: tryb wstawiania, wyświetlanie wierszy informacyjnych, kontrola pisowni na bieżąco, upływ czasu, liczba uderzeń w klawisze, po których automatycznie jest wykonywany zapis (*Autosave*), jednostka długości.
Nawet gdy program jest w rzeczywistości wykorzystywany przez jedną osobę, to takie zasady współpracy z T³ ułatwiają hierarchizację i podział tematyczny pracy, np. przez automatyczne montowanie różnych wolumenów i plików sekwencji klawiszowych.

Wszystkie obiekty z poziomu głównego mają swojego właściciela i prawa dostępu obejmujące możliwości: czytania, poprawiania, zmiany nazwy, kopiowania, przenoszenia, usuwania i zmiany tych praw.

W T³ istnieje pojęcie użytkownika uprzywilejowanego (ang. *master user*). Tylko on jest uprawniony do modyfikowania fontów, wprowadzania i zmiany parametrów innych użytkowników, zmiany właściciela i praw dostępu do obiektów należących do innych użytkowników.

Licząca kilkaset stron dokumentacja programu T³ [16] obejmuje krótkie wprowadzenie, podręcznik dla początkujących, podręcznik referencyjny oraz dokumentację techniczną. Niewielka demonstracja możliwości programu – uwzględniająca definiowanie nowej klawiatury, pisanie tekstu matematycznego i chemicznego – została przygotowana w postaci pliku sekwencji klawiszowych.

Przygotowywanie tekstów w języku polskim

Polskie litery zostały uwzględnione w firmowo przygotowanym zestawie znaków o nazwie *Latin Accents*, co oznacza, że są widoczne na ekranie oraz na wydruku dla wszystkich sterowników graficznych i drukarek uwzględnionych przy implementacji programu T³ (od 1988 r.). Przykładowy tekst w języku polskim (słowa piosenki „Góralu ...") znalazł się nawet w dokumencie demonstracyjnym.

Znaki z zestawu *Latin Accents* są dostępne w sześciu fontach (*Elite, Pica,* 17, *Slant,* 17 *Slant*). Niezbyt wielkim nakładem pracy można rozszerzyć tę grupę o inne odmiany, np. o *Latin Accents* 10, przydatny przy pisaniu streszczeń artykułów, nagłówków czy tekstów wielołamowych.

Pomimo pewnej zbieżności nazwy *Latin Accents* z nazwą IBM Latin 2, pozycje polskich liter nie odpowiadają ani tej, ani żadnej innej konwencji polskich liter przyjętej w stosowanych w Polsce rozwiązaniach (Mazovia, Microvex itd.) [7]. Nie stanowi jednak problemu napisanie we własnym zakresie krótkiego programiku wykonującego konwersję z jakiegokolwiek standardu stosowanego dotychczas, na konwencję przyjętą w zestawie znaków *Latin Accents,* powiększoną o 128. W kolejnym kroku, otrzymany w wyniku takiej konwersji tekstowy plik ze znakami z rozszerzonego zestawu ASCII można wczytać do dokumentu T³ z zachowaniem wystąpień polskich liter – wystarczy wtedy wybrać font IBM dla znaków ASCII o kodach 0–127 i font *Latin Accents* dla znaków o kodach 128–255.

Klawiaturę zawierającą polskie litery można przygotować na dwa sposoby. Rozwiązanie pierwsze polega na utworzeniu nowej klawiatury przez umieszczenie polskich liter w miejsce ich łacińskich odpowiedników (*ą* na miejscu *a*, *Ą* na miejscu *A* itd.) i dołączeniu jej np. jako klawiatury uzupełniającej (dostępnej przez ⟨Alt⟩).

Rozwiązanie drugie polega na rozszerzeniu istniejącej już klawiatury (np. podstawowej *Same as keys*) z wykorzystaniem klawiszy nieaktywnych. Rezygnując z aktywności, powiedzmy, klawisza kreski ukośnej ⟨/⟩ definiujemy, że naciśnięcie (kolejno po sobie) klawiszy ⟨/⟩ i ⟨a⟩ spowoduje wprowadzenie litery *ą*, klawiszy ⟨/⟩ i ⟨A⟩ – litery *Ą* itd. Ten mechanizm korzystania z rozszerzeń klawiatury przypomina stosowany w programie ChiWriter sposób wprowadzania jednego znaku nie należącego do aktualnego fontu (np. ⟨F2⟩ + ⟨a⟩).

Jako uzupełnienie do zaprojektowanych klawiatur można potraktować odpowiednio zdefiniowane sekwencje klawiszowe, zamieniające w wyróżnionym tekście znaki z różnych odmian zestawu *Latin Accents* między sobą – np. *Latin Accents* na *Latin Accents Slant* i odwrotnie. Decyzja o tym: ile i jakie klawiatury oraz sekwencje klawiszowe należy sobie przygotować, zależy od metody pracy preferowanej przez użytkownika.

Sprawdzenie poprawności pisowni tekstów odbywa się z użyciem dwóch słowników: wbudowanego oraz dodatkowego słownika użytkownika w zwykłym pliku ASCII. Wyrazy zawie-

rające litery niełacińskie są podczas tej kontroli pomijane. Kolejne wersje T³ mają umożliwić uzupełnianie słowników o wyrazy pisane z użyciem innych alfabetów.

Dzielenie i przenoszenie polskich wyrazów jest wykonywane zgodnie z ogólną metodą wspomnianą przy omawianiu dokumentów.

Istnieje łatwy sposób podmiany tekstów wyświetlanych przez *Help* na ich polskie odpowiedniki (jednak bez polskich liter), choć celowość takiego postępowania jest dyskusyjna.

T³ a T_EX

Profesor J. Milne, autor publikacji [8], oceniał możliwości współdziałania z T_EX-em czterech programów (dla IBM PC) do przetwarzania tekstów technicznych, próbujących wypełnić lukę między dwoma ideałami: wprowadzaniem w konwencji WYSIWYG i otrzymywaniem wydruku z użyciem T_EX-a. Wspomnianymi programami były: **T³** (wersja 2.3 z 1990 r.), **ChiWriter** (3.17), *EXP* (2.0) i **Leo** (1.1). Wewnętrzną postacią dokumentu tworzonego przez program Leo jest plik źródłowy T_EX-a. Pozostałe trzy programy zapewniają możliwość wykonania konwersji własnych dokumentów na postać akceptowaną przez T_EX-a. W chwili powstawania artykułu [8] gotowy do rozpowszechaniania był tylko konwerter opracowany dla programu T³. Pozostałe dwa były na ukończeniu (testowe wersje beta).

Możliwości podanych wyżej programów zostały sprawdzone m.in. na złożonych, wyeksponowanych (ang. display) formułach matematycznych zawartych w dokumencie demonstracyjnym programu T³ (2.21). Interesujący nas konwerter przekształcił bezbłędnie kilka tych wzorów, z pozostałymi natomiast poradził sobie częściowo. W większości przypadków konwerter T³ potrafił również prawidłowo zidentyfikować i przekształcić zależności matematyczne wkomponowane w zwykły tekst (ang. *in-line mathematical expressions*).

Trudności w skonstruowaniu konwertera wynikają z dwu podstawowych przyczyn. Po pierwsze – dokument T³ nie zawiera informacji o strukturze złożonych wyrażeń matematycznych, a jedynie opisuje położenie ich poszczególnych symboli składowych względem linii bazowej. Po drugie – T³ nie ma specjalnego trybu matematycznego, tak jak T_EX.

Plik źródłowy otrzymywany w wyniku konwersji jest dostosowany do wymagań bazowego (ang. *plain*) T_EX-a i jego zasobu fontów. Istnieje jednak możliwość określenia, w pliku sterującym pracą konwertera, reguł konwersji dodatkowych fontów, takich jak *Fraktur* czy *Blackboard bold,* akceptowanych np. przez AMS-T_EX-a.

Oceniając globalnie przydatność konwertera T³, Milne stwierdził, że zaoszczędza on około 80–90% nakładu pracy potrzebnej do przekształcenia dokumentu T³ na plik źródłowy T_EX-a.

T³ a CHIWRITER

Obydwa te programy przeznaczone do redagowania tekstów technicznych są do siebie zbliżone z funkcjonalnego punktu widzenia (WYSIWYG, teksty wielojęzyczne, formuły wielopoziomowe itd.). Różnią się jednak zasadniczo organizacją programu i zasadami współpracy z użytkownikiem. Zdaniem autora wspomnianego już artykułu [8], T³ pretenduje przy tym do roli programu najwyższej jakości w grupie WYSIWYG TWP, ChiWriter natomiast kładzie główny nacisk na prostotę posługiwania się.

Milne wyróżnił kilka **cech przemawiających na korzyść ChiWritera:**

- lepsza organizacja menu,
- łatwiejsze kopiowanie tekstu między dwoma dokumentami (dwa okna),
- redagowanie fragmentów formuł wielopoziomowych co najmniej tak dobrze dopracowane jak w T³, a łatwiejsze w użyciu,
- opcje wygodnego rysowania linii i prostokątnych ramek,
- (zapowiedziane w wersji 4.) rozszerzające się symbole pierwiastków i dużych nawiasów.

Cechy przemawiające za T³ to:

- automatyczne formatowanie wierszy tekstu (planowane w wersji 4 ChiWritera),
- dostępność dla każdego z wprowadzonych do tekstu znaków jego odmian – pogrubienie, podkreślenie lub przekreślenie (ChiWriter – tylko standardowe fonty i ograniczone kombinacje tych atrybutów),
- bardziej rozbudowane możliwości formatowania dokumentów (regiony z graficzną prezentacją ich rozmieszczenia, „biblioteki" formatów wierszy i stron itd.),
- lepsza jakość wydrukowanych dokumentów (starannie zaprojektowane fonty proporcjonalne), większa elastyczność automatycznego doboru odstępu między wierszami (ang. *line spacing*).

Podobieństwa:

- sposób konstruowania (składania) złożonych wyrażeń matematycznych,
- bardzo duże podobieństwo (pod względem możliwości) wersji dystrybucyjnej konwertera ChiWriter → T_EX do konwertera T³ → T_EX.
- bardzo dobra dokumentacja do obu programów.

Podsumowując swoją ocenę, Milne stwierdził, że obydwa programy: T³ i ChiWriter dają się łatwo poznać i skutecznie używać, jakość otrzymywanego wydruku jest adekwatna do większości potrzeb, a ich konwertery na postać T_EX-a wykonują dużą część koniecznej pracy. ChiWriter jest programem prostszym do nauki, ale brak mu kilku bardziej wyrafinowanych cech T³.

Z mojego, subiektywnego oczywiście, punktu widzenia na korzyść posługiwania się programem T³ (w porównaniu do ChiWritera 3.xx) przemawiają jego bogatsze i wygodniejsze możliwości formatowania tekstu, lepiej dopracowany aparat sekwencji klawiszowych, różnorodność i jakość przygotowanych fontów dające – moim zdaniem – lepsze efekty w postaci produktu końcowego, jakim jest wydruk dokumentu.

Odpowiadają mi również: idea tworzenia biblioteki dokumentów wzorcowych, stanowiąca naturalną konsekwencję sposobu użytkowania T³, samowystarczalność każdego dokumentu oraz, z ogólniejszych aspektów, łatwe przystosowywanie T³ do indywidualnych wymagań użytkownika i rodzaju przygotowywanych dokumentów.

Na koniec wreszcie, istotne znaczenie ma dla mnie fakt korzystania w jednolity sposób z wbudowanych w T³ możliwości redagowania zarówno formuł matematycznych, jak i tekstów w języku polskim (ChiWriter nawet w wersji 4.01 z 1991 r. nie ma wbudowanych polskich fontów), angielskim lub rosyjskim.

* * *

Z informacji zawartych w literaturze wynika, że firma TCI przygotowuje implementację programu T³ działającą pod kontrolą systemu operacyjnego **XENIX.** Należy oczekiwać, że w ten

sposób poszerzony zostanie jeszcze bardziej, i tak już duży, krąg użytkowników tego dobrego programu TWP.

LITERATURA

[1] Barron D., Rees M.: Text Processing and Typesetting with UNIX. Addison-Wesley, 1987
[2] Bień J.: Co to jest TeX. Wiadomości Matematyczne XXIX pp. 131–156, 1990
[3] Boston Computer Society PC Technical Group: Technical Wordprocessors for the IBM PC and Compatibles. Notices of AMS, Vol. 33, No. 1, pp. 8–37, 1986
[4] Brown E.: AmiPro 2.0: Looking Good. PC World, pp. 93–94, August 1991
[5] Furuta R., Scofield J., Shaw A.: Document Formatting Systems: Survey, Concepts, and Issues. Computing Surveys, Vol. 14, No. 3, pp. 417–472, September 1982
[6] Goldstein R., Loomis J., Tekewsky A.: Technical Wordprocessors for the IBM PC and Compatibles. Report by the Boston Computer Society. Part I – TWP Capabilities and People Needs. Notices od AMS, Vol. 34, No. 1 pp. 15–32, January 1987; Part IIA – TWP Summary Tables, Notices of AMS, Vol. 34, No. 2 pp. 262–281, February 1987; Part IIB – Reviews, Notices of AMS, Vol. 34, No. 3, pp. 462–491, March 1987
[7] Majewski W.: Z komputerem po polsku.. Komputer, nr 10, 1987
[8] Milne J.: Four Word Processors with T_EX Capabilities. Notices of AMS, Vol. 37, No. 8, pp. 1018–1022, October 1990
[9] Palais R.: Mathematical Text Processing. Notices of AMS Vol. 33, No. 1, pp. 3–7, 1986
[10] Palais R.: Mathematical Text Processing. Notices of AMS, Vol 35, No. 3, pp. 391–396, March 1988
[11] Scavo T.: Technical Writing with Word 4.0. Notices of AMS, Vol. 36, No. 10, pp. 1353–1358, December 1989
[12] Scavo T.: Math Writer 2.0 A Software Review. Notices of AMS, Vol. 38, No. 6, pp. 568–572, July–August 1991
[13] Seymour J. et al.: Scientific Word Processors: Formulas for Success: PC Magazine, Vol. 7, No. 13, pp. 251–328, May 29, 1988
[14] Siebenmann L.: Toward Wider Use of T_EX Typesetting. A Pre-T_EX Manifesto. Notices of AMS. Vol. 33, No. 4, pp. 597–607, April 1986
[15] Stone M.: The Business of Words: Scientific. PC Magazine No. 5, pp. 185–198, February 26, 1986
[16] T^3 Scientitic Word Processing System Documentation: Reference, Technical Reference, Quick-start Guide, Tyro Tutorial. TCI Software Research, New Mexico, 1988
[17] Wallstrom T.: The Equation Processor in Word 3.0. Notices of AMS, Vol. 35, No. 2, pp. 263–265, February, 1988.

Piąta Wiosenna Szkoła PTI

Szczecińskie Koło Polskiego Towarzystwa Informatycznego i jego Sekcja Informatyki Stosowanej w Zarządzaniu organizują w dniach od 18 do 22 maja br. Piątą Wiosenną Szkołę PTI. Podobnie jak w roku ubiegłym, impreza odbędzie się w Ośrodku Szkoleniwo-Wypoczynkowym „DOLNA ODRA" w Świnoujściu. Tematem tegorocznej Szkoły będzie „Metodologia tworzenia systemów informatycznych w CASE".

Genezą Wiosennych Szkół PTI była myśl zorganizowania forum wymiany poglądów środowiska profesjonalnego na temat metodologii oraz metod i technik tworzenia systemów informatycznych. Piąta edycja Szkoły, stanowiącą reasumpcję tematyki Szkół wcześniejszych, będzie poświęcona wymianie poglądów na współczesne metodologie tworzenia SI oraz nowym komputerowym narzędziom wspomagania inżynierii oprogramowania. Komitetowi organizacyjnemu Szkoły w składzie: *prof. Tadeusz Wierzbicki, dr Zdzisław Szyjewski* oraz *mgr Mirosław Frąckiewicz* udało się pozyskać jako referentów przedstawicieli wiodących firm i instytucji z zakresu tematyki Szkoły. Firma IBM przedstawi koncepcję AD/Cycle a Digital – koncepcję tworzenia systemów otwartych. Interesujące będą również wystąpienia *Witolda Staniszkisa*, z firmy ZETO-RODAN, który omówi doświadczenia wspomagania procesu tworzenia systemu, oraz *Stanisława Wryczy*, z Uniwersytetu Gdańskiego, którego referat „Tutorial – metody i techniki tworzenia SI", obejmujący wykład i ćwiczenia, będzie próbą zorganizowania zajęć praktycznych w grupach. Istotnym uzupełnieniem tematyki obrad będzie przygotowany przez firmę CSBI referat omawiający doświadczenia stosowania bazy danych Progress, co umożliwi wymianę poglądów również na temat stosowania języków czwartej generacji. Organizatorzy zamierzają uzupełnić wykłady i ćwiczenia pokazami ciekawszych rozwiązań obecnie oferowanych na rynku narzędzi projektowania SI.

Dodatkowe informacje na temat Szkoły można uzyskać w Kole PTI **w Szczecinie tel. 760-31 w. 317 lub 319.**

BOGDAN MARZEC
VMEpro
Kraków

OS-9 – siła i słabość standardu

Niniejszy artykuł stara się naświetlić – w nawiązaniu do niedawnych publikacji w INFORMATYCE – wysuwające się na czoło aspekty problematyki systemów czasu rzeczywistego [6, 9]. Spostrzeżenie główne, to silny trend do standaryzacji i otwartości systemów zarówno w zakresie sprzętu, jak i oprogramowania. Konstatacja ta poznawczo bałaby banalna, gdybyśmy ją umieli przełożyć w Polsce na język praktyki. Za to, że tak się nie dzieje, trudno już dzisiaj winić politycznie motywowane dawne bariery dla transferu zaawansowanych technologii na Wschód. Raczej powinniśmy wytknąć sami sobie, że w spadku po czasach miniowych pozostał nam brak umiejętności urzeczywistniania zasady prymatu efektywności. To ona bowiem stanowi ekonomiczne zaplecze postępującej standaryzacji i otwartości systemów czasu rzeczywistego.

Wśród otwartych architektur komputerowych tworzących współczesną bazę sprzętową dla systemów czasu rzeczywistego umacnia się wiodąca rola standardu VMEbus. Tworzący ją zbiór reguł i właściwości (Pierwsza kodyfikacja, tzw. Revision A, pod którą podpisało się konsorcjum firm Motorola, Mostek i Signetics, pochodzi z 1981 roku), pozytywnie przeszedł próbę czasu, najpierw sprawnie udostępniając rosnące możliwości mikroprocesorów rodziny 680 × 0, a ostatnio architektur typu RISC. Ponadto sformalizowana, jednoznacza definicja standardu z 1985 r. tzw. Revision C, która została objęta zarówno amerykańską normą IEEE, jak i międzynarodową IEC, zapewniła rzeczywistą zgodność elektryczną i mechaniczną produktów różnych firm. Obecnie oferta rynkowa sprzętu VME jest bogata i kompleksowa, jakość produktów bardzo dobra, a ceny niewygórowane. Od strony programowej standardowi VMEbus towarzyszy wiele systemów i programów zarządzających (ang. *executive*) czasu rzeczywistego. Firmy, które je oferują, nie ujawniają danych handlowych. Niemniej następujące fakty:

- OS-9 [1] jest dostępny na zdecydowanej większości płyt jednostek centralnych VME/680 × 0,
- producenci pozostałych płyt (ze specjalizowanymi wejściami-wyjściami, przetwornikami A-C i C-A, sterownikami itp.) z reguły wyposażają je w procedury obsługi (ang. *device driver*) zgodne z systemem OS-9,
- ilość niezależnego oprogramowania (ang. *third party software*), rozszerzającego ofertę firmy Microware, m.in. o kompilatory C i C++, interpreter poleceń (ang. *shell*) Bourne'a, systemy okien zgodne i wzorowane na X-Windoes, bije na głowę to, czym może pochwalić się konkurencja,
- istnieje liczna kadra programistów, którzy poznali OS-9 już w czasie studiów lub w trakcie prowadzonych przez wiele firm specjalistycznych kursów,

uzasadniają tezę, iż OS-9, choć nie wsparty żadną normą, jest de facto standardem (lub inaczej – standardem przemysłowym – ang. *industry standard*) wśród systemów operacyjnych czasu rzeczywistego.

Co to oznacza dla firmy wyspecjalizowanej we wdrożeniach systemów czasu rzeczywistego? W typowych przypadkach nie traci ona czasu ani pieniędzy na wykonanie jeszcze jednego systemu operacyjnego czasu rzeczywistego, nie tworzy też własnego oprogramowania narzędziowego i stosuje gotowe procedury obsługi płyt. Do jej obowiązków należy natomiast stworzenie koncepcji najwłaściwszego rozwiązania konkretnego problemu i odpowiednie do niej zintegrowanie „cegiełek" sprzętowych i programowych.

Ta metodyka postępowania odbiega daleko od tego, za czym opowiadają się zwolennicy komputerowej autarkii. Na poparcie takiego stanowiska można wysuwać następujące argumenty:

- obszar zastosowań czasu rzeczywistego, gdzie w pełni wykorzystuje się możliwości obliczeniowe mikroprocesorów 16-, a zwłaszcza 32-bitowych, jest mocno ograniczony,
- integrowanie gotowych modułów sprzętowych i programowych przy użyciu uniwersalnych narzędzi o znacznym stopniu ogólności (m.in. języków wysokiego poziomu, a nawet narzędzi CASE) prowadzi do powstania sysemów mało efektywnych, ustępujących systemom specjalizowanym, których każdy element został ukształtowany pod kątem potrzeb specyficznego zastosowania,
- mikrokomputer jednoukładowy jest zdecydowanie tańszy niż np. płyta jednostki centralnej opartej na MC = 68030, a ceny zachodniego oprogramowania systemowego i narzędziowego idą w setki, a nawet tysiące dolarów.

Odpowiadając na te argumenty nie wystarczy poprzestać na wygłoszeniu opinii, że zachodnie podejście zostało pozytywnie zweryfikowane przez rynek, do którego wymagań również Polska będzie musiała się wkrótce dostosować. Można bez trudu wykazać, że często spotykana nadwyżka mocy nowszych generacji mikroprocesorów w zastosowania czasu rzeczywistego (gdy mamy na myśli samą realizację algorytmu obliczeniowego) faktycznie nie idzie na marne. Jest ona spożytkowywana na to, by z nawiązką zażegnać realną groźbę mniejszej efektywności systemu modularnego w porównaniu z systemem specjalizowanym opartym na prostym mikrokomputerze jednoukłado-

[1] Microware Systems Corporation. 1900 N.W. 114th Street, Des Moines, Iowa, USA

Mgr inż. **BOGDAN MARZEC** absolwent Wydziału Elektrotechniki, Automatyki i Elektroniki Akademii Górniczo-Hutniczej w Krakowie. Po studiach pracował w Zakładzie Metrologii AGH, a następnie w Instytucie Obróbki Skrawaniem w Krakowie. Przez ostatnie dwa lata przebywał na kontrakcie zagranicznym. Od bieżącego tego roku pracuje w firmie VMEpro w Krakowie, zajmującej się systemami sterowania w czasie rzeczywistym.

wym. Umożliwia także obsłużenie wyrafinowanego graficznego interfejsu użytkownika (ang. *Graphical User Interface* – GUI), którego jakość stała się jednym z głównych kryteriów, jakimi kierują się w swych decyzjach potencjalni nabywcy. Ponadto posługiwanie się gotowymi, dobrze przetestowanymi narzędziami i „cegiełkami" przynosi zdecydowane skrócenie cyklu tworzenia (ang. *time-tomarket*) systemu czasu rzeczywistego, czemu dodatkowo towarzyszy zwiększenie jego niezawodności.

Standardami przemysłowymi rzadko stają się rozwiązania najciekawsze pod względem technicznym lub najbardziej zaawansowane technologicznie. Na gruncie elektroniki użytkowej dobitnie tego dowodzi zwycięstwo standardu VHS nad Betamax; w świecie mikrokomputerów wystarczy wskazać architekturę PC i DOS. Analogicznie jest ze standardami opartymi na powszechnie uznawanych normach – tu konieczność pogodzenia zapatrywań różnych grup interesów oraz negocjacyjny sposób dochodzenia do końcowych rozstrzygnięć stanowią dość skuteczną tamę dla propozycji zbyt nowatorskich, za daleko odbiegających od status quo. Standardem staje się zazwyczaj mocny średniak, wypróbowany i niezawodny, sprawdzający się w większości sytuacji. I taki też jest OS-9. Warto więc bliżej się z nim zapoznać.

Modularność systemu

System czasu rzeczywistego OS-9 jest oparty na koncepcji modułów pamięciowych. Został zaprojektowany w taki sposób, by każdy moduł realizował określone funkcje. Ta modularność umożliwia włączanie poszczególnych modułów do systemu operacyjnego lub ich usuwanie w trakcie konfigurowania OS-9 dla potrzeb konkretnego komputera. Przykładowo, OS-9 rezydujący w pamięci ROM małego systemu sterowania nie potrzebuje tych wszystkich modułów, które obsługują pamięci masowe.

Modularna struktura OS-9 ułatwia stosowanie modularnych technik programowania i pozwala łatwo dopasować go do potrzeb użytkownika. Do skonfigurowania systemu nie jest potrzebne ponowne skompilowanie całego systemu operacyjnego, a tylko określonego modułu pamięciowego zapewniającego realizację dodatkowej usługi, Co więcej, konfigurację można przeprowadzić w trakcie funkcjonowania systemu.

Modularność umożliwia instalowannie OS-9 w komputerach o zróżnicowanych stopniach złożoności. Wersja przemysłowa (Industrial OS-9) zapewnia realizację funkcji systemowych najistotniejszych dla systemu czasu rzeczywistego, rezyduje w pamięci EPROM i jest przeznaczona dla bardzo wydajnych zastosowań. Wersja profesjonalna (Professional OS-9) stanowi natomiast kompletny, wyposażony w dyskowe i taśmowe pamięci masowe oraz możliwości pracy w sieci, wielodostępny system tworzenia oprogramowania.

OS-9 dopuszcza kilka różnych rodzajów modułów. Każdy z nich ma inne zastosowanie i spełnia inną funkcję. Moduły nie muszą być kompletnymi programami ani nawet nie muszą być napisane w języku programownia. Muszą jednak spełniać dwa zasadnicze kryteria: przesuwności i wielobieżności. Pierwsze z nich wynika z tego, że moduły pamięciowe OS-9 są wywoływane przez nazwę, a nie przez adres, w związku z czym mogą być przechowywane w dowolnym miejscu w pamięci. Wielobieżność umożliwia odwoływanie się do tego samego modułu przez dwa lub więcej procesów jednocześnie, co zwiększa efektywność wykorzystania pamięci.

Podstawowymi grupami modułów systemu OS-9 są:
- jądro, moduł zegara obsługujący konkretną realizację sprzętową zegara czasu rzeczywistego oraz moduł INIT, znajdujący zastosowanie przy inicjalizacji systemu,
- zarządcy plików (ang. *file manager*), których zadaniem jest obsługa operacji wejścia-wyjścia dla wyróżnionych klas podobnych urządzeń zewnętrznych,
- procedury obsługi (ang. *device drivers*) obsługujące fizyczną realizację operacji wejścia-wyjścia przez konkretny typ sterownika urządzeń zewnętrznych,
- deskryptory urządzeń (ang. *device descriptor*) – niewielkie tablice zawierające informacje pomocnicze, niezbędne do realizacji operacji wejścia-wyjścia.

Jądro czasu rzeczywistego

Funkcje systemowe. Jądro czasu rzeczywistego OS-9 zarządza pamięcią i procesorem oraz oferuje liczne usługi systemowe. OS-9 dopuszcza dwa tryby wykonywania programów wynikowych: tryb użytkownika (ang. *user state*) i tryb systemowy (ang. *system state*). Tryb użytkownika to normalne środowisko programowe, w którym odbywa się wykonywanie procesów. Tryb systemowy jest tym stanem, w którym są wykonywane wywołania systemowe i procedury obsługi przerwania. Usługi systemowe są realizowane przez zbiór wywołań, z których każde należy do jednej z następujących kategorii:
- wywołania systemowe trybu użytkownika – zarządzają pamięcią i procesami pozostając w związku z wykonywaniem programów użytkowych i abstrahując od specyficznych właściwości sprzętu komputerowego;
- wywołania systemowe trybu systemowego – są wykonywane tylko w trybie nadrzędnym (ang. *supervisor*) mikroprocesora 680 × 0; nie mogą być używane przez programy trybu użytkownika i zazwyczaj działają bezpośrednio na sprzęcie komputerowym zapewniając reakcje w czasie rzeczywistym na zdarzenia zachodzące w świecie zewnętrznym;
- wywołania systemowe wejścia-wyjścia – realizują rozmaite funkcje wejścia-wyjścia i są przetwarzane przez zarządców plików i procedury obsługi odpowiednie dla danego urządzenia. To podejście odciąża jądro czasu rzeczywistego i zapewnia dużą elastyczność OS-9. Wszystkie wywołania systemowe wejścia-wyjścia mogą być używane przez programy zarówno trybu systemowego, jak i trybu użytkownika.

Inicjalizacja. Jądro czasu rzeczywistego OS-9 jest odpowiedzialne za inicjalizację systemu po wyzerowaniu sprzętu oraz za wykonanie procedury startowej. Inicjalizacja odbywa się na podstawie danych zapisanych w module INIT. Jest to moduł niewykonywalny, zawierający specyficzne dla danej konfiguracji systemu parametry w jednoznacznie zdefiniowanych kolejnych polach.

Zarządzanie pamięcią. Jądro czasu rzeczywistego OS-9 dokonuje dynamicznego przydziału pamięci systemowej oraz pamięci dla procesów. Podczas realizacji procedury startowej OS-9 wykrywa obszary dostępnej pamięci (za pomocą następujących po sobie zapisów i odczytów komórek pamięci, wybranych w kolejnych obszarach o wielkości 8 KB). Następnie, korzystając z tak utworzonej mapy pamięci i jej zasobu, przydziela procesom pamięć według algorytmu „first-fit". OS-9 prowadzi rejestr wszystkich modułów znajdujących się w pamięci. Każdy zapis w tym rejestrze zawiera adres modułu i licznik procesów używających modułu. OS-9 umożliwia rozpoznawanie różnych typów pamięci i przeznaczanie ich do specjalnych celów – np. pamięć ekranu lub pamięć o zawartości podtrzymywanej zasilaniem bateryjnym.

Zarządzanie procesami. Jądro czasu rzeczywistego OS-9 umożliwia współbieżne wykonywanie programów. Każdy program w systemie OS-9 jest wykonywany jako proces, a każdy proces może zapewnić sobie dostęp do pożądanych zasobów systemu

przez odpowiednie wywołania systemowe. OS-9 jest ponadto odpowiedzialny za ładowanie zadań, ich inicjalizację i wykonanie. Praca z podziałem czasu umożliwia procesom wspólne korzystanie z procesora. OS-9 wykorzystuje priorytetowy program szeregujący pracujący okrężnie (ang. *round robin*), który przydziela czas procesora dla żadego procesu. W każdej chwili proces jest w jednym z trzech stanów:

- aktywności, w którym proces jest aktywny i gotowy do wykonania,
- oczekiwania, w którym proces jest nieaktywny, dopóki proces-potomek nie zostanie zakończony lub nie nadejdzie sygnał adresowany do niego,
- uśpienia, w którym proces jest nieaktywny przez określony czas lub do momentu nadejścia sygnału.

Stanom tym odpowiadają trzy kolejki deskryptorów procesów. Zmiana stanu procesu polega na przeniesieniu jego deskryptora do innej kolejki. Kolejka procesów aktywnych jest sortowana według wieku. Wiek procesu jest to suma początkowego priorytetu, z jakim proces pojawił się w kolejce, oraz liczby przełączeń procesów, które od tego momentu wystąpiły. Stan procesu i jego priorytet początkowy mogą być określane zarówno przez system, jak i przez użytkownika. Kolejny kwant czasu (ang. *time-slice*) otrzymuje proces najstarszy.

OS-9 udostępnia następujące cztery mechanizmy synchronizacji i komunikacji między procesami:

- moduły danych, które mają standardowy format modułów pamięciowych, co zapewnia ich przesuwność; są dostępne do czytania i zapisywania w dowolnym momencie dla wszystkich procesów mających odpowiednie prawa dostępu;
- **potoki** (ang. *pipe*) i **potoki oznaczone** (ang. *named pipe*), umożliwiające komunikację procesom współbieżnym na zasadzie wykorzystania bufora FIFO (ang. *first in first out*). Dane wyjściowe z jednego procesu są wejściem dla drugiego procesu, przy czym wielkość bufora FIFO wynosi 90 bajtów. Dwa procesy mogą komunikować się przez więcej niż jeden potok; proces może też wysyłać przez potok dane do samego siebie;
- **zdarzenia** (ang. *events*), różniące się od modułów danych i potoków tym, że przekazują informację sterującą między dwoma procesami, same natomiast nie wymieniają danych. Jeśli istnieje zasób systemowy, z którego korzysta kilka procesów współbieżnych, to zdarzenia umożliwiają synchronizację dostępu tych procesów do zasobu. Zdarzenie w OS-9 to zarządzana przez system struktura danych o długości 32 bajtów, zawierająca m.in. numer identyfikacyjny, nazwę, wartość zdarzenia i wartość inkrementu dodawanego do wartości zdarzenia;
- **sygnały** (ang. *signals*), które są znacznie mniejsze niż zdarzenie i w danym momencie mogą być przesyłane tylko między dwoma procesami. Są używane do sterowania wielu asynchronicznie wykonywanych procesów, a także do ochrony współużytkowanych zasobów. Każdy sygnał zawiera 16-bitowy kod umożliwiający jego identyfikację. Proces może sekwencyjnie wysłać sygnały do wielu procesów, może też otrzymywać sygnały z więcej niż jednego źródła. Proces otrzymujący sygnał musi, pod groźbą przerwania, zawierać procedurę obsługi sygnału. W procedurze tej jest zakodowana reakcja procesu na sygnał.

Użytecznym narzędziem wprowadzania uzależnień czasowych są alarmy. W trybie użytkownika wywołanie systemowe F$Alarm umożliwia programowi wysłanie sygnału do samego siebie o określonej porze lub po upływie określonego czasu. Sygnał taki może być też wysyłany periodycznie. W trybie systemowym upływ czasu alarmu powoduje coś więcej – wykonanie przez jądro określonego podprogramu z najwyższym priorytetem.

Mechanizmy czasu rzeczywistego. System OS-9 zawiera następujące mechanizmy czasu rzeczywistego:

- wywłaszczalne przełączanie zadań, umożliwiające reakcję w czasie rzeczywistym przez wywłaszczenie wykonującego się właśnie procesu, gdy proces o wyższym priorytecie stanie się aktywny. Wywłaszczony proces traci resztę swego kwantu czasowego i powraca do kolejki procesów aktywnych. Oznacza to, że proces o najwyższym priorytecie jest wykonywany natychmiast;
- nadzorowanie wykonywaniem procesu – to specjalne wywołanie systemowe F$SetSys umożliwiające dostęp do dwóch globalnych zmiennych systemowych. D_MinPty definiuje minimalny priorytet, poniżej którego wiek procesów nie zwiększa się ani nie są one brane pod uwagę przy szeregowaniu. Ustawienie D_MinPty na określonym poziomie sprawia, że wszystkie procesy znajdujące się poniżej są wstrzymane, dopóki nie zakończą się pewne krytyczne czasowo zadania. D_MaxAge określa największy wiek, do którego procesy mogą się „starzeć". W rezultacie wszystkie procesy zostają zaklasyfikowane do jednej z dwóch grup: o niskim lub wysokim priorytecie. Procesy z grupy o niskim priorytecie przestają się „starzeć" na poziomie D_MaxAge, co oznacza, że procesy z grupy o wysokim priorytecie otrzymują cały czas procesora;
- obsługa przerwań to instalacja nowej procedury obsługi przerwania w systemie dokonywana przez wywołanie systemowe F$Irq. Za jego pośrednictwem system operacyjny rejestruje adres początkowy procedury obsługi przerwania, numer wektora, jaki podaje urządzenie będące źródłem przerwania, fizyczny adres tego urządzenia, jego priorytet w ramach wektora oraz wskaźnik do przydzielonego obszaru pamięci globalnej. W OS-9 pojawienie się przerwania nie oznacza automatycznego wykonania określonego procesu. Zamiast tego procedura obsługi przerwania może podjąć akcję, która zmieni stan procesu na aktywny; może też ustawić flagę, której stan monitoruje odpowiedni proces. Szybkiej i efektywnej reakcji na przerwania sprzyja uczynienie procesu, z którym komunikuje się przerwanie, wykonywanym w trybie systemowym. Wówczas ma on dostęp do wszystkich instrukcji mikroprocesora i wszystkich zasobów sprzętowych z pominięciem wszelkich czasochłonnych zabezpieczeń. Zmiana kontekstu dla takiego procesu odbywa się szybciej, gdyż nie trzeba przechodzić z trybu użytkownika do trybu nadrzędnego. Taki proces ma też dostęp bez ograniczeń do wywołań jądra czasu rzeczywistego, w tym do mechanizmu szeregowania zadań.

Zarządzanie wejściem-wyjściem. System OS-9 charakteryzuje się modularnym, zunifikowanym, niezależnym od specyfiki sprzętowej systemem wejścia-wyjścia, który można rozbudowywać lub konfigurować. W systemie wejścia-wyjścia stosuje się notację ścieżki do pliku wzorowaną na UNIX-owej.

Przykładowo:

ścieżka do pliku dyskowego	:/h0/⟨skorowidz⟩/.../⟨nazwa pliku⟩
ścieżka do portu szeregowego	:/t1
ścieżka do pamięci taśmowej	:/mt0
ścieżka w sieci	:/n0/⟨nazwa węzła⟩/⟨urządzenie⟩/⟨skorowidz⟩

Zdecydowana różnica w stosunku do UNIX-a polega natomiast na tym, że OS-9 odciąża jądro z realizacji większości przetwarzania związanego z operacjami wejścia-wyjścia, przerzucając je na barki zarządcy plików. Reakcje w czasie rzeczywistym ułatwia to, że system wejścia-wyjścia jest sterowany przerwaniami i ukierunkowany na przetwarzanie strumienia pojedynczych bajtów. Typowy proces wejścia-wyjścia można przedstawić następująco:

Żądanie danej (status)	**UŻYTKOWNIK**	**Otrzymanie danej (status)**
↓		↑
Identyfikacja zadania jako dotyczącego wejścia-wyjścia. Identyfikacja potrzebnych zasobów: zarządcy plików programu obsługi i pamięci. Przekazanie zadania do zarządcy plików.	**Jądro czasu rzeczywistego OS-9**	Jądro współpracuje z zarządcą plików w przekazaniu danej (statusu) do użytkownika.
↓		↑
Sprawdzenie poprawności zadania. Realizacja przetwarzania niezależnego od urządzenia wejścia-wyjścia. Wywołanie programu obsługi do obsłużenia sprzętu.	**Zarządca plików**	Zarządca plików koordynuje zwrot danej (statusu)
↓		↑
Wejście w interakcje ze sprzętem i wykonania faktycznego transferu danych.	**Procedura obsługi urządzenia**	Procedura obsługi zwraca daną (status) do zarządcy plików.

W OS-9 zarządcy plików są modułami wielobieżnymi, co pozwala użyć jednego zarządcy do obsługi całej klasy urządzeń mających podobne właściwości operacyjne. Jest ich pięć:
SCF (*Sequential Character File Manager*) – jest używany do obsługi urządzeń znakowych, takich jak porty szeregowe i równoległe;
RBF (*Random Block File Manager*) – jest używany do obsługi urządzeń blokowych o dostępie bezpośrednim, jak dyski stałe i dyskietki;
PIPEMAN (*Pipe File Manager*) umożliwia komunikację międzyprocesową z wykorzystaniem potoków;
SBF (Sequential Block File Manager) jest używany do obsługi urządzeń blokowych o dostępie sekwencyjnym, jak pamięć taśmowa;
NFM (*Network File Manager*) łączy w sposób przezroczysty systemy wejścia-wyjścia wielu systemów OS-9.

Procedury obsługi urządzeń wykonują podstawowe funkcje wejścia-wyjścia na poziomie sprzętu, przy czym wielobieżny pojedynczy moduł może jednocześnie obsługiwać wiele urządzeń, o identycznym typie sterownika. Moduł procedury obsługi składa się z siedmiu procedur, wywoływanych przez zarządcę plików i realizowanych w trybie systemowym.

Deskryptory urządzeń to małe, niewykonywalne moduły udostępniające informacje, które kojarzą określone urządzenie wejścia-wyjścia z jego nazwą logiczną, fizycznym adresem sterownika, nazwami odpowiedniego zarządcy plików i procedury obsługi oraz danymi inicjującymi. W odróżnieniu od zarządców plików i programów obsługi, deskryptory urządzeń nie mają ogólnego charakteru, co oznacza, że każdemu deskryptorowi odpowiada tylko jedno urządzenia wejścia-wyjścia (chociaż danemu urządzeniu wejścia-wyjścia można przypisać kilka deskryptorów, różniących się np. danymi inicjalizacyjnymi).

Interpreter poleceń

System czasu rzeczywistego OS-9 dysponuje wzorowanym na UNIX-ie interpreterem poleceń (ang. *shell*), który umożliwia użytkownikowi dostęp do wewnętrznych funkcji systemu operacyjnego. Poleceń tych jest ponad siedemdziesiąt i pozwalają one m.in. ustalać środowisko pracy użytkownika, przydzielać pamięć, wykonywać operacje na plikach, zmieniać przyporządkowanie standardowych urządzeń wejścia-wyjścia, redagować teksty.

Rezydentne środowisko tworzenia oprogramowania

OS-9 oferuje rozbudowane i kompletne rezydentne środowisko do tworzenia oprogramowania. Składają się na nie kompilatory języków C, Pascal, Fortran, interakcyjny kompilator Basica, makroasembler. Dostępne są trzy programy uruchomieniowe: trybu systemowego, trybu użytkownika i poziomu źródłowego języka C, a także edytor pełnoekranowy, pakiet komunikacyjny, spooler drukarki i pakiet poczty elektronicznej.

Skrośne środowisko tworzenia oprogramowania

UniBridge. Jest to pakiet programowy umożliwiający tworzenie oprogramowania w języku C i asemblerze oraz komunikację między systemem UNIX-owym, a systemem OS-9. Umożliwia wykorzystanie systemu UNIX we wszystkich fazach tworzenia oprogramowania. Bazę sprzętową dla UniBridge stanowi sieć Ethernet, a komputerami macierzystymi mogą być m.in. stacje robocze Sun 3 i Sun 4, DEC VAX, HP 9000 Seria 300.

PCBridge. Część PCBridge rezyduje w komputerze osobistym z systemem DOS, a część w komputerach z systemem OS-9. Połączenie jest zrealizowane przez interfejs szeregowy. Programy użytkowe są przygotowywane na PC, a następnie ładowane do systemu docelowego do testowania lub wykonania.
VMS/VAX. Dla komputerów VAX pracujących pod kontrolą systemu VMS jest dostępny kompilator skrośny języka C.

OS-9 a konkurencja

Zagrożenia dla pozycji OS-9 są szczególnie widoczne w następujących dziedzinach:
- zastosowania czasu rzeczywistego o najwyższych wymaganiach. W Niemczech coraz większą popularnością w tej dziedzinie cieszy się jądro czasu rzeczywistego pSOS$^+$.

Następujące porównanie:

	Czas przełączania procesora (kontekstu)	Opóźnienie w przyjęciu zgłoszenia (przerwania)
OS-9 (68020,20 MHz)	55.1 + 1.5t μs*	11.1 μs**
pSOS$^+$ (68020,25 MHz)	19 μs	6 μs

gdzie: * t reprezentuje liczbę zadań w kolejce procesów aktywnych;
** przy założeniu, że danemu wektorowi przerwania odpowiada tylko jedna procedura obsługi przerwania.

dokończenie na s. 28

System autorski ASYS 3.0 jest zrealizowany w ramach projektu badawczego nr 8-0275-91-01 finansowanego przez KBN.

DARIUSZ GAIK
JACEK KOTULA
JAN PIECHA
Zakład Elektroniki i Informatyki
Uniwersytet Śląski
Katowice

System autorski ASYS 3.0

Wysiłki badawcze ostatnich lat w dziedzinie informatyki dotyczą w dużej mierze graficznych narzędzi projektowania. Specyficzną grupę zastosowań komputera stanowi oprogramowanie dydaktyczne, rozwijane od wielu lat, a stanowiące daleko zaawansowaną dziedzinę badawczą. Istotnym krokiem na drodze poszukiwań „przyjaznego" oprogramowania upraszczającego proces opracowania programów dydaktycznych są tzw. systemy autorskie. W artykule omówiono system narzędzi graficznych opracowany w Zakładzie Elektroniki i Informatyki Uniwersytetu Śląskiego – ASYS 3.0.

Jedną z miar jakości oprogramowania narzędziowego czy użytkowego jest jego przyjazność. Termin ten oznacza łatwość przyswajania jego treści przez użytkownika. Nowe generacje języków jak C++, Turbo Pascal 6.0 i inne, coraz bardziej efektywne i łatwiejsze w obsłudze, ale są ciągle językami ogólnego przeznaczenia. Najważniejszym problemem dla użytkownika komputera jest kwestia jak najkrótszego czasu pozyskania oprogramowania użytkowego lub otrzymania efektywnego narzędzia, które pozwoli mu opracować samodzielnie potrzebny program.

Systemy autorskie (ang. *Authoring Systems*) zawierają zestaw narzędzi oprogramowania. Są one istotnym krokiem na drodze poszukiwań oprogramowania upraszczającego proces opracowania programów dydaktycznych. Użytkownik operuje między innymi: interaktywnym edytorem graficznym, edytorem tekstów, narzędziami animacji obrazu, edytorem dźwięku, konwersacyjną bazą danych oraz importerami obrazów i dźwięków analogowych (techniką multimedialną). Technika menu (ang. *menu driven system*) prowadzi projektanta przez kolejne szczeble projektowania, eliminując konieczność opanowania trudnej sztuki programowania.

Na wstępie dyskusji na temat narzędzi symulacji koniecznym jest określenie istotnych różnic między następującymi trzema podstawowymi klasami oprogramowania dydaktycznego:
CAI (ang. *Computer Assisted Instruction*), określające programy prezentacyjne, w których konwersacja z komputerem nie jest sprawą istotną; są one bowiem używane przez nauczyciela jako element ilustracji zajęć,
CBT (ang. *Computer Based Teaching*), określające programy używane również przez nauczyciela o znacznych możliwościach rozwidleń kursu z elementami konwersacji z komputerem,
CAL (ang. *Computer Aided Learning*), określające programy przeznaczone do samodzielnego uczenia się, wyposażone w bogaty system konwersacji i wartościowania odpowiedzi uczącego się. Zamiast CAL jest stosowana również nazwa **ITS** (ang. *Intelligent Tutoring System*, – inteligentny system uczący) wykorzystujący tzw. bazę wiedzy. Na podstawie kolejnych etapów sprawdzania poziomu opanowania przekazywanego materiału są automatycznie wybierane dalsze fazy kształcenia (rozwidlenia tematyczne, poziom szczegółowości itp.).

Niestety większość autorów publikacji na temat zastosowań komputerów w nauczaniu nie rozróżnia wymienionych wyżej pojęć. Omówienie cech oraz analiza przydatności narzędzia programowania winna być poprzedzona określeniem zadań, jakie za ich pomocą pragniemy wykonać. Ogólną klasyfikację systemów kształcenia wspomaganego komputerem w kategoriach programowanego nauczania przedstawiono w pracach [2, 3, 4].

Charakterystyka systemu ASYS

ASYS 3.0 jest systemem autorskim opracowanym w latach 1989–1990 w Zakładzie Elektroniki i Informatyki Uniwersytetu Śląskiego [1]. Jest to obiektowo-zorientowany zestaw narzędzi projektowania programów demonstracyjnych CAT. Projektowanie materiałów CAI odbywa się dwupoziomowo (techniką buttom-up); poziom niższy – opracowanie ekranów i obiektów bibliotecznych, poziom wyższy – etap konsolidacji elementów programu.

Niski poziom programowania

Jest to etap edycji graficznej ekranów, stanowiących elementarną jednostkę programu. Niski poziom programowania jest reprezentowany programem EDYTOR, zawierającym komendy obsługi bazy danych (biblioteki obiektów i ekranów) oraz narzędzia edycji. Każdy z obiektów jest reprezentowany zbiorem definiowalnych atrybutów, takich jak: kolor, font, rozmiar, prędkość animacji i inne. W każdym obiekcie można wyróżnić kilka warstw taktowanych komendą „naciśnij klawisz". Formy graficzne mogą być uzupełnione elementami animacji i elementem interakcji „naciśnij klawisz". Opcja EDYTOR jest podstawowym elementem projektowania materiałów CAI z edytorem grafiki i animacji, edytorem tekstów, modułem animacji, edytorem dźwięków, programatorem funkcji (dla prostych obliczeń) oraz programem uruchomieniowym („debagerem") opracowanego obiektu.

Wysoki poziom programowania

Jednostki zostają złożone w program za pomocą konsolidatora pakietu. Złożenie tworzy menu odpowiadające nazwom obiektów. Na etapie konsolidacji określamy dodatkowe elementy interakcji. Na zakończenie każdego ekranu można wywołać ekran następny lub, klawiszem Esc, menu zawierające listę nazw ekranów. Odpowiednia manipulacja konsolidatorem zapewnia

prostą interakcję w ramch opracowanego bloku programowego. W pakiecie ASYS 3.0 nie przewidziano narzędzi oceny odpowiedzi, a więc zaawansowanej interakcji.

Wymagania sprzętowe:

Proste PC – 12 MHz XT lub AT, 640 kB RAM, nie jest konieczny dysk sztywny, karta z trybem EGA, DOS 3.x.

Instalacja ASYS 3.0

Pakiet ASYS może pracować na komputerach PC – XT/AT-286/386 z dowolną barwną kartą graficzną i jednym dyskiem elastycznym. Minimalny wymiar pamięci operacyjnej wynosi 520 kB. Praca z komputerem w takiej konfiguracji jest możliwa choć uciążliwa ze względu na konieczność częstej zamiany dyskietki systemowej na biblioteczną. Ta niedogodność sugeruje celowość użycia dysku sztywnego.

System ASYS 3.0 jest uruchamiany bezpośrednio z dysku, bez potrzeby uprzedniego zainstalowania. Uruchamianie systemu odbywa się za pomocą programu zarządzającego ASYS.exe przez zlecenia o tej samej nazwie.

Zlecenia i instrukcje edytora

Pakiet ASYS zgłasza się planszą menu głównego zawierającego trzy podstawowe opcje: EDYTOR, KONSOLIDATOR i WYJŚCIE do systemu operacyjnego DOS. W każdej opcji systemu rezydują dwa podstawowe tryby pracy. W opcji EDYTOR wyróżniamy tryby: BAZA i EDYCJA, natomiast w opcji KONSOLIDATOR – tryby: JEDNOSTKA i NLOK.

Tryb BAZA

KATALOG – ustawienie ścieżki dostępu plików edycyjnych,
CZYTAJ – wczytanie pliku edycyjnego z katalogu do bufora pamięci systemu,
DOPISZ – dopisywanie do bufora systemowego pliku z katalogu edycyjnego,
ZAPISZ – zapisywanie zawartości bufora pamięci do katalogu,
USUŃ – wyczyszczenie bufora pamięci.

Tryb EDYCJA

GRAFIKA – menu zarządzające procedurami graficznymi:
Kropka – rysowanie punktu w miejscu położenia kursora graficznego,
Linia – rysowanie linii łamanej o odcinkach wyznaczanych przez kolejne położenia kursora graficznego. Kursor jest „zaczepiony" w końcu poprzedniego odcinka linii łamanej,
Ramka – rysowanie ramki prostokątnej (nie wypełnionej) o przekątnej wyznaczonej położeniem kursora graficznego,
Prostokąt – rysowanie prostokąta (wypełnionego bieżącym kolorem „pióra") o przekątnej wyznaczonej położeniem kursora graficznego,
Kolor – ustawienie koloru „aktywnego", którym wykonywane będą zlecenia graficzne i tekstowe,
Wypełnienie – rysowanie wypełnienia obszaru zamkniętego zadanym kolorem,
Okrąg – rysowanie okręgu, którego środek i długość promienia wyznacza się kursorem graficznym,
Gumka – wymazywanie dowolnej części aktualnego ekranu graficznego,
Kopiowanie – powielanie dowolnych obszarów aktualnego ekranu graficznego,
Obrót – obrót o 90° dowolnych obszarów ekranu graficznego objętych ramką edycyjną,

TEKST – zlecenia edytora tekstu:
Wysokość – jest deklaracją jednej z czterech wysokości tekstu,
Krój – jest deklaracją jednego z czterech kształtów liter,
Pisz – jest komendą przejścia do edycji tekstu,

ANIMACJA – menu zarządzające procedurami animacji:
Ruch – generowanie płynnego ruchu elementu po zadanym torze,
NK – jest deklaracją zatrzymania programu i kontynuacji po naciśnięciu dowolnego klawisza,
Dźwięk – programowanie efektów dźwiękowych (prosty edytor melodyczny),
Pauza – jest deklaracją zatrzymania wykonywania programu przez zadany czas.

FUNKCJA – zlecenie obliczania i wykreślania charakterystyki funkcji jednej zmiennej, złożonej z funkcji elementarnych:
- potęgowej,
- wykładniczej,
- logarytmicznej,
- trygonometrycznej,
- hiperbolicznej.

W deklaracji funkcji określa się w kolejności:
- miejsce na ekranie, w którym ma się pojawić funkcji (okno),
- wzór funkcji (w oknie),
- krańce przedziału zmian wartości zmiennej x, odłożonej na osi odciętych układu współrzędnych,
- zakres wartości funkcji: $y = f(x)$.

Przykład

Deklaracja funkcji trygonometrycznej: $y(x) = \sin x + x^2$

Funkcja:....$y(x) = \sin x + x \star x$ ⟨Enter⟩,
Insert X range = ...0,6 ⟨Enter⟩, (od 0 do 6 rd).
Insert Y range = ...0,36 ⟨Enter⟩.

Wynikiem tak zadeklarowanej funkcji będzie wykres złożenia jednego niepełnego okresu funkcji $\sin(x)$ i paraboli.

FAZY – zlecenia umożliwiające przeglądanie i kontrolę wykonywania redagowanego ekranu (program uruchomieniowy systemu):

Wykonaj – uruchomienie ciągu instrukcji programu zapisanych w buforze pamięci. Wykonanie rozpoczyna się od następnej instrukcji, po ostatnio wykonanej, lub od początku bufora (patrz zlecenie „Od początku"). Realizacja instrukcji jest zatrzymywana w punkcie stopu lub przy ostatniej instrukcji zapisanej w buforze pamięci. Jeśli użytkownik wygrał śledzenie w trybie pracy krokowej, zlecenie kończy się po wykonaniu jednej instrukcji. Jeżeli bufor programu jest pusty lub wskaźnik instrukcji wskazuje na koniec bufora, zlecenie jest pomijane.
Od Początku – ustawienie wskaźnika instrukcji na początek bufora programu. Ekran monitora jest zerowany.
Krok – ustaw-skasuj, umożliwia śledzenie realizacji programu w trybie pracy krokowej. Tryb pracy krokowej jest sygnalizowany gwiazdką w munu pola **Krok**.
Stop – ustaw-skasuj stop programowy (odpowiednik ang. *break point*) w programie przez ostatnio wykonaną instrukcją.
Stopy usuń – umożliwia usunięcie wszystkich wcześniej zadeklarowanych punktów stopów.
Instrukcja usuń – umożliwia usunięcie z programu zbędnej instrukcji ostatnio zapisanej lub ostatnio wykonywanej (w pracy krokowej). Jednorazowo można usunąć tylko jedną instrukcję. Wykonanie zlecenia nie powoduje usunięcia z ekranu tej części rysunku, która była efektem wykonania usuniętej instrukcji – aby sprawdzić efekt skasowania trzeba ponownie uruchomić redagowany fragment programu.

Tryb OBIEKT

Umożliwia deklarowanie biblioteki typowych elementów składowych programów użytkownika.
KATALOG OBIEKTÓW – służy do ustawnienia ścieżki dostępu do obiektów.
ZAPISZ OBIEKT – jest opcją umożliwiającą deklarację ekranu lub zaznaczenie jego części i zapisywanie tak zdefiniowanego obiektu do poprzednio zdefiniowanego katalogu. Zapisany zostaje element nowego zbioru bibliotecznego.
CZYTAJ OBIEKT – jest opcją umożliwiającą odczyt obiektu z listy obiektów zapisanych w katalogu bibliotecznym. Obiekt można wstawić w dowolne miejsce na ekranie. Zlecenie jest dopisywane do zawartości bufora edycyjnego.

Środowisko edytora

JĘZYK KOMEND. Aby przyspieszyć dostęp użytkownika do poszczególnych opcji edytora oraz usprawnić „poruszanie się" w ich obrębie przygotowano zestaw sterujących komend, wywołujących niezależnie od aktywnego menu dowolne zlecenia edytora. Słownik tego języka wyświetlany jest po naciśnięciu klawiszy *^K (Crtl K)*.

POMOCNIK EDYTORA. Do dyspozycji użytkownika zaprojektowano system pomocnika – podpowiadacza, opisujący działanie wszystkich opcji edytora oraz ich sposób wywołania. Wywołanie pomocnika odbywa się za pomocą klawiszy *^H (Crtl H)* i dotyczy bieżącej opcji menu systemu ASYS.

Konsolidator systemu

Scalanie tworzonych za pomocą EDYTORA części programu CAI w jedną całość odbywa się za pomocą KONSOLIDATORA, w dwu etapach:
I – konsolidacji jednostki i II – konsolidacji bloku.

Tryb JEDNOSTKA

Opcja **JEDNOSTKA** jest pierwszym etapem konsolidacji programu do postaci użytkowej. Polega ona na logicznym uporządkowaniu utworznych uprzednio ekranów. Proces przebiega w dwóch etapach:
a) użytkownik wpisuje nazwy używanych ekranów ustalając jednocześnie ich kolejność,
b) użytkownik redaguje menu jednostki (specjalny ekran generowany automatycznie przez system); jest to lista nazw kolejnych ekranów programu użytkownika, które można wywołać w dowolnej chwili.

Etap konsolidacji jednostki polega na wypełnieniu planszy przedstawionej na wydruku. W podświetlonych polach wpisujemy kolejno:
- ścieżkę dostępu do katalogu, w którym znajdują się scalane pliki ekranów,
- ścieżkę dostępu do katalogu, w którym ma zostać zapisany plik wynikowy dla redagowanej jednostki,
- nazwę redagowanej jednostki,
- nazwę pliku ekranu zgłoszenia (ekranu, który pojawi się przy wywołaniu jednostki jako jej czołówka);
- nazwy plików ekranów składowych jednostki.

Wszystkie nazwy plików ekranów podaje się bez rozszerzeń (nie więcej niż 8 znaków). W przypadku błędu zapisu pojawia się komunikat: *Brak zbioru lub błędna nazwa zbioru!*

Nazwy plików nie mogą się powtarzać. W przypadku powtórzenia nazwy ukazuje się komunikat: *Podwójna deklaracja nazwy zbioru!*

```
Edycja jednostki

Podaj ścieżkę dostępu  do plików ekranów :    C:\ASYS\EDYCJA
Podaj ścieżkę dostępu do pliku jednostki :    C:\ASYS\PRZYKLAD
Podaj nazwę jednostki: JEDN-1.

Podaj nazwy plików ekranów :

Ekran zgłoszenia :  EKR-0
 1: EKR-1      4: EKR-4      7: EKR-7     10: EKR-10
 2: EKR-2      5: EKR-5      8: EKR-8     11:
 3: EKR-3      6: EKR-6      9: EKR-9     12:

F1 - Pomoc   F2 - Katalog   F9 - Kontynuacja   F10 - Wyjście
```

Formularze konsolidacji jednostki

Komentowane są również inne błędy programisty, a ponadto zabezpieczono program przed takimi pomyłkami operatorskimi, jak nieopatrzne skasowanie pliku, przejście do niewłaściwej opcji itp.

Klawisze funkcyjne reprezentują następujące opcje:
F1 – Pomoc – użytkownik uzyskuje podstawowe informacje o sposobie redakcji formularza jednostki,
F9 – umożliwia wyjście z fazy redagowania ekranu menu jednostki,
F10 – zaniechanie wykonywania opcji i powrót do głównego menu pakietu ASYS.

Po zakończeniu edycji „ekranu menu" wyszczególnione pliki ekranów są łączone w jeden plik jednostki.

Tryb BLOCK

W tym trybie uzyskuje się gotowy program użytkowy. Przebieg edycji oraz znaczenie klawiszy jest identyczne jak w opisanym wcześniej etapie edycji jednostki. Jedyna różnica polega na tym, że do formularza konsolidacyjnego bloku wpisujemy nazwy jednostek. Po zakończeniu edycji bloku (po naciśnięciu klawisza F9) następuje konsolidacja, w wyniku której powstaje plik bloku. Program ten jest automatycznie kompilowany wraz z konsolidacją plików pomocniczych. W wyniku tej operacji uzyskuje się program w postaci wykonawczej (wynikowej) – ★ *exe*, uruchamiany bezpośrednio z poziomu systemu operacyjnego DOS.

Plik źródłowy systemu

Wynikiem pracy z edytorem systemu ASYS 3.0 jest zbiór poleceń systemowych, zapisywanych w buforze edytora, a po wykonaniu zlecenia ZAPISZ (w trybie BAZA) jest generowany w katalogu bibliotecznym plik źródłowy, jako ciąg wywołań procedur systemowych zawartych w bibliotece PROC. typu. Doświadczony użytkownik systemu może opracowywać i poprawiać źródła systemowe pod dowolnym edytorem tekstu (w kodach ASCII).

Kierunki rozwoju systemu

Dla bardziej rozbudowanych materiałów CAI (kilkadziesiąt ekranów) może okazać się niewystarczającym maksymalny rozmiar jednego pliku, redagowanego za pomocą ASYS. W nowej wersji pakietu etap konsolidacji będzie generował inną strukturę organizacyjną. Dotychczasowy samodzielny plik wykonawczy zostanie zamieniony na plik zarządzający poszczególnymi plikami źródłowymi (ekranami).

dokończenie na s. 23

AMT ACCEL-535

– to najszybszy i najbardziej wszechstronny drukarko-ploter

Intelli-Plot TURBO

Przygotuj się do nowego, fascynującego sposobu kreślenia projektów przy pomocy Intelli-Plot Turbo produkowanego przez AMT (Advanced Matrix Technology). Wzięliśmy w pełni wyciszoną, pracującą na formacie do A2, drukarkę AMT ACCEL-535, dodaliśmy język graficzny Hewlett-Pacarda (HP-GL) oraz procesor AutoCAD ADI. Rezultatem jest najszybszy i najbardziej wszechstronny drukarko-ploter.

Intelli-Plot Turbo dzięki szeregowym i równoległym interfejsom może być łączony szybko i bez wysiłku z PC, mini, mainframe albo wybraną stacją CAD. Po prostu prześlij rysunki ze swojego niezależnego programu do Intelli-Plot Turbo, a on je wykreśli szybciej niż jakikolwiek ploter DTP. Przy tym nie potrzeba żadnych plików pośrednich, dodatkowych programów zajmujących pamięć stałą, ani dodatkowego hardware'u.

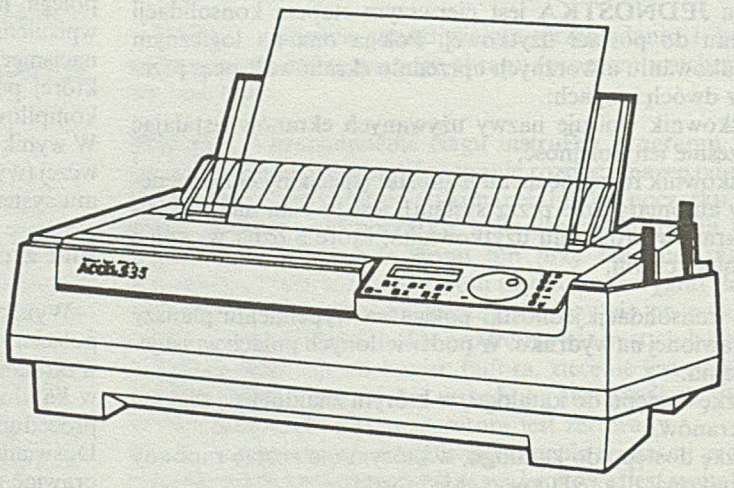
Intelli-Plot Turbo proponuje Ci rozszerzoną listę możliwości, których nie znajdziesz w ploterach DTP, zaczynając od tego, że ma różne możliwości (szybkość, jakość) kreślenia; niskojakościowe do wydruku kontrolnego, jakość średnia oraz dokładniejsze, wysokojakościowe do wyjątkowo przejrzystych i precyzyjnych wykresów.

Intelli-Plot Turbo ma także wbudowaną możliwość drukowania aż do 20 rysunków jednocześnie, w żądanej kolejności. Istnieje też możliwość zmian w Twoim projekcie bez ponownego użycia komputera, powiększanie lub zmniejszanie w nieprawdopodobnej skali, przesuwanie i wybieranie fragmentów kopiowania, skalowanie automatyczne i ustawianie w żądanym formacie. Możesz wybierać z palety 16 kolorów, 3 grubości linii dla każdego z 15 pisaków.

Intelli-Plot drukuje na różnych rodzajach papieru włącznie z pergaminem. Do ładowania papieru możesz używać wbudowanego podajnika, podajnika dodatkowego z możliwością ładowania papieru w formacie do A2 albo podawać papier z roli.

DANE TECHNICZNE:

- **emulacja HP-GL HP-7475 oraz Epson, IBM, DEC etc.**
- **MTBF 15.000 godz.**
- **format A2**
- **druk w 16 kolorach**



Advanced Matrix Technology Inc.

KUPON Inf. 5/92

Zainteresowanych dodatkowymi informacjami związanymi z naszą ofertą, prosimy o nadsyłanie wypełnionego kuponu pod adresem: **ABC Data – 00-865 Warszawa, ul. Wallców 13.**

Imię i Nazwisko .....

Zakład pracy, stanowisko .....

.....

ul. .....

Kod ..... Miasto .....

ABC DATA

Przedstawicielstwo w Polsce:
00-865 Warszawa
ul. Wallców 13
tel. 24-11-43, 24-78-35
fax 24-12-83, tlx 825098

31-066 Kraków
ul. Skawińska 11
tel. 21-98-92
fax 21-96-80, tlx 326497

358/92

Atrapa mechanizmów wyjątków dla C

Mechanizm wyjątków powstał jako uzupełnienie programowania strukturalnego. Początkowo nie spotkał się z dobrym przyjęciem. Głośny był swego czasu artykuł C. Hoare'a [2] zawierający ostry atak na wprowadzenie przez Ichbiaha wyjątków do Ady. Niechęć do mechanizmu wyjątków u zwolenników „czystego" programowania struktualnego wynikała prawdopodobnie z nadziei na rychłe wypracowanie praktycznych technik dowodzenia poprawności programów. Techniki te opierały się na bezpośredniej odpowiedniości struktur syntaktycznych i przepływu sterowania. Niedopuszczalna więc była nie tylko instrukcja **goto**, ale wszystko co tę odpowiedniość łamało.

Teorie dowodzenia poprawności programów nie spełniły jednak pokładanych w nich nadziei, a coraz bardziej widocznym było, że trzymanie się zasad „czystego" programowania strukturalnego – wbrew temu co twierdzili jego zwolennicy – bynajmniej nie prowadzi do czytelnych programów. Wskazywał na to Knuth w artykule [5] już w 1974 roku. Każdy nieco bardziej złożony program, obok normalnego właściwego sobie algorytmu, musi uwzględnić sytuacje specjalne: przepełnienie tablic, uszkodzenie urządzeń zewnątrznych, ingerencje użytkownika i in. Według niektórych statystyk tylko 10% programu zajmuje się jego zasadniczym algorytmem, a reszta to kontrola poprawności argumentów, interakcja z użytkownikiem i obsługa różnych zdarzeń szczególnych. Bez dodatkowych mechanizmów sterowania zasadniczy algorytm programu utonie w tym wszystkim i cały program stanie się mało czytelny.

Niezależnie od wyznawanych teorii życie miało swoje wymagania i np. twórcy pierwszych kompilatorów Pascala – mimo deklarowanej niechęci do instrukcji **goto** – pozostawili ją w języku, a nawet byli zmuszeni do wprowadzenia np. takiego pokracznego rozwiązania jak *etykieta ratunkowa* (zob. [4]), aby w miarę sensownie obsłużyć błędy czytania danych. W języku C, który posłużył nie tylko do napisania kompilatora samego siebie, ale również całego systemu operacyjnego UNIX, mamy obok lokalnych instrukcji **goto** również tzw. **długie skoki**, tj. skoki pomiędzy różnymi poziomami wywołań funkcji. Jest to konsekwentnie zaprojektowany mechanizm realizowany za pomocą dwóch funkcji systemowych **setjmp** i **longjmp**. Najpierw trzeba zadeklarować blok do zmagazynowania opisu środowiska wywołania (rys. 1). Struktura tego bloku o nazwie **jmp_buf** jest opisana w pliku nagłówkowych **setjmp.h.**

```
#include <setjmp.h>

. . .
jmp_buf SWEET_HOME;
```

Rys. 1

Opis środowiska powrotu jest ładowany za pomocą funkcji **setjmp**, która po wykonaniu swojego zadania zwraca wartość zerową. Długi skok realizuje się za pomocą funkcji **longjmp**, wołanej z dwoma parametrami: adresem bloku środowiska powrotu i kodem powrotu (rys. 2). Ten ostatni musi być różny od zera. Powrót z funkcji **longjmp** następuje w to samo miejsce co powrót z odpowiadającego wywołania **setjmp**, ale z wartością równą drugiemu argumentowi funkcji **longjmp**.

```
switch(setjmp(SWEET_HOME)) {

   case 0: break;

      . . .
   case 1:

      . . .
   default:

      . . .
}

   . . .
longjmp(SWEET_HOME, 1);

   . . .
```

Rys. 2

Funkcje **setjmp/longjmp** można za pomocą odpowiednich konstrukcji językowych odpowiednio ubrać i nadać im postać nowoczesną: właśnie mechanizmu wyjątków. Poniżej przedstawiam rozwiązanie, które od pewnego czasu z powodzeniem stosuję. Rozwiązanie jest wzorowane na mechanizmach Ady.

```
#include <setjmp.h>

typedef struct {      /*exception block format*/
   jmp_buf *_xx_jbp;  /*jump block pointer*/
   int _xx_id;        /*exception identifier*/
} EXCEPTION;

#define handle(exno) jmp_buf _xx_jmbf;\
   struct { EXCEPTION _xx_exception,\
      *_xx_prev; } _xx_save[exno];\
   int _xx_ret, _xx_cnt=0, _xx_loop;\
   if((_xx_ret=setjmp(_xx_jmbf)), 0) { ;
#define when(exc) }else if((_xx_ret==0)?\
   (_xx_save[_xx_cnt]._xx_exception=exc, \
   _xx_save[_xx_cnt]._xx_prev=&(exc),\
   exc._xx_jbp=_xx_jmbf, exc._xx_id=(++_xx_cnt), 0):\
   (_xx_ret==exc._xx_id)) { ;
#define raise(exc) longjmp(exc._xx_jbp, exc._xx_id)
#define restore() for(_xx_loop=_xx_cnt; _xx_loop--;) {\
   *(_xx_save[_xx_loop]._xx_prev)=\
      _xx_save[_xx_loop]._xx_exception; }
```

Rys. 3

Rysunek 3. przedstawia definicje wszystkich obiektów, które skadają się na całe rozwiązanie. Mamy tu definicję typu **EXCEPTION** oraz cztery makrodefinicje: **handle, when, raise, restore.** Typ **EXCEPTION** to struktura definiująca wyjątek. Do zgłaszania wyjątku służy makroinstrukcja **raise**. Obsługę wyjątków, czyli *handler*, buduje się za pomocą makroinstrukcji **handle** i **when**. Instrukcją **restore** przywraca się poprzednią obsługę wyjątków. Rysunki 4. i 5. są ilustracją zastosowania.

Rysunek 4. przedstawia pakiet funkcji **DigAndDrill**. Funkcje pakietu reagują na trzy zdarzenia, które zgłaszają jako trzy wyjątki: **AlarmStatus, ExternInterrupt, DeviceMalfunction**. Wyjątki te są zdefiniowane w pakiecie jako struktury zewnętrzne. Zgłoszenia wyjątku dokonuje się za pomocą makroinstrukcji **raise**. Zgłoszenie wyjątku powoduje przerwanie normalnej sekwencji sterowania i przejście do odpowiedniego handlera wyjątków. W dynamicznym łańcuchu wywołań funkcja zgłaszająca wyjątek musi być poprzedzona przez co najmniej jedną funkcję zawierającą handler zgłaszanego wyjątku lub sama zawierać taki handler. Zgłoszenie jest przechwytywane

przez najbliższy handler (w dynamicznym łańcuchu wołań) zawierający obsługę tego wyjątku. Konstrukcję handlerów ilustruje rys. 5.

```
/* DigAndDrill package */

EXCEPTION AlarmStatus, ExternInterrupt, DeviceMalfunction;

void Dig()
{
    . . .
  raise(ExternInterrupt);

    . . .
  raise(AlarmStatus);

    . . .
}

void Drill()
{
    . . .
  raise(ExternInterrupt);

    . . .
  raise(DeviceMalfunction);

    . . .
}
```

Rys. 4

Rysunek 5. przedstawia pewien program korzystający z pakietu **DigAndDrill**. Program główny woła funkcję **Works**, a ta z kolei woła funkcje pakietu: **Dig** oraz **Drill**. Funkcje te w reakcji na zachodzące zdarzenia zgłaszają wyżej wspomniane wyjątki. Zarówno funkcja **Works** jak i program główny zawierają handlery wyjątków. Handlery umieszcza się na początku części instrukcyjnej funkcji (inaczej niż np. w Adzie). Handler jest zbudowany z wołań dwóch makroinstrukcji: **handle** i **when**. Handler ma strukturę „grzebieniową": jego skrajne „zęby" to makrowołanie **handle** i zamykający nawias klamrowy. Argumentem makrowołania **handle** jest liczba obsługiwanych wyjątków. Wewnętrzne „zęby" handlera są wyznaczane przez wołania makroinstrukcji **when**. Argumentem makrowołania **when** jest wyjątek. Po tym makrowołaniu następuje sekwencja dowolnych instrukcji języka C. Tej sekwencji przekazuje się sterowanie, gdy zostanie zgłoszony odpowiedni wyjątek.

Bieg sterowania w handlerze jest inny, gdy handler jest inicjowany i inny gdy zostanie zgłoszony wyjątek. W pierwszym przypadku handler zapisuje dane organizujące obsługę, natomiast wcale nie wykonuje żadnych z instrukcji pomiędzy „zębami" handlera. W drugim przypadku wykonują się instrukcje po „zębie" odpowiadającym zgłoszonemu wyjątkowi.

Funkcja **Works** zawiera handler obsługujący dwa wyjątki: **ExternInterrupt** i **AlarmStatus**. Nie obsługuje natomiast wyjątku **DeviceMalfunction**, mimo że funkcja **Drill** może zgłosić taki wyjątek. W związku z tym powinien on być obsłużony na wyższym poziomie (w sensie dynamicznego następstwa wywołań funkcji). Przyjrzyjmy się teraz bliżej handlerowi w funkcji **Works**. Normalnym wyjściem z obsługi wyjątku jest przejście do pierwszej instrukcji za handlerem. Jeżeli więc obsługa wyjątku **ExternInterrupt** nie zawiera niesekwencyjnego wyjścia, po jej zakończeniu funkcja **Works** zacznie wykonywać się od początku.

Obsługa wyjątku **AlarmStatus** kończy się przywróceniem poprzedniej obsługi i zgłoszeniem wyjątku **DeviceMalfunction**. Należy pamiętać, że każde opuszczenie funkcji zawierającej handler powinno być poprzedzone przywróceniem poprzedniej obsługi wyjątków.

```
#include "DigAndDrill"

Works()
{
  handle(2)
    when(ExternInterrupt)
        . . .
    when(AlarmStatus)
        . . .
      restore();
      raise(DeviceMalfunction);
  }

  Dig();
    . . .
  Drill();
    . . .
  restore();
}

main()
{
  handle(1)
    when(DeviceMalfunction)
      printf( ... );
      exit(5);
  }
    . . .
  Works();
    . . .
}
```

Rys. 5

W związku z wprowadzeniem powszechnego podatku dochodowego i koniecznością informowania właściwych Urzędów Skarbowych o wszystkich wypłatach zwracamy się do Autorów wysyłających teksty do opublikowania o podawanie następujących danych:

- **nazwisko, imiona (pierwsze i drugie),**
- **imiona: ojca i matki,**
- **miejsce i data urodzenia,**
- **numer identyfikacyjny PESEL (wpisany przez Biuro Meldunkowe do dowodu osobistego – nie mylić z numerem dowodu osobistego!),**
- **dokładny adres (miejsce zameldowania),**
- **adres Urzędu Skarbowego właściwego dla miejsca zamieszkania Autora (bardzo ważne – bez tej informacji nie możemy przekazać honorarium do wypłaty!).**

Prosimy także – do naszej wiadomości – podać numer telefonu służbowego i ustalić formę przekazania honorarium (kasa Wydawnictwa, poczta, konto).

Program główny ma handler tylko jednego wyjątku **Device-Malfunction**. Jego obsługa to jedynie wypisanie komunikatu i wyjście z programu. Wyjątek ten jest zgłaszany przez funkcję **Drill**, wołaną przez funkcję **Works**, lub przez handler funkcji **Works**.

Wyjątek może być obsługiwany przez handler tej samej funkcji, przez którą został zgłoszony. Ten sam wyjątek może być obsługiwany przez różne handlery na różnych poziomach. Handlery mogą więc sobie przekazywać obsługę wyjątków. Konieczne jest, aby każdy zgłaszany wyjątek był obsługiwany przez jakiś handler na tym samym lub dowolnym, wyższym poziomie.

Zaletą proponowanego rozwiązania jest łatwość jego zastosowania. Nie jest wymagany jakiś specjalny preprocesor, ani żadna biblioteka funkcji. Wystarczy przepisać definicje z rys. 3. Wadą jest natomiast ograniczoność tak skonstruowanego mechanizmu; nie można np. wyspecyfikować obsługi dla listy wyjątków lub dla „wszystkich pozostałych" (*others*). Przykra jest również konieczność pamiętania o wszystkich zależnościach składniowych i semantycznych, co normalnie jest zadaniem kompilatora lub preprocesora. Niektórych tych wad nie ma rozwiązanie Allmana i Beena [1], ale wymaga ono sporej biblioteki skomplikowanych funkcji, częściowo kodowanych w assemblerze. A zapewne najlepszym rowiązaniem jest poczekać na odpowiednio rozwinięty język C++.

LITERATURA

[1] Allman E., Been d.: An Exception Handler for C. W *Proc. Summer 1985 USENIX Conference*. Portland, May 1985

[2] Hoare C. A. R.: The emperor's old clothes. 1980 Turing Award Lecture. W *Commun ACM 24*, 2 February 1981

[3] Ichbiah J. D.: *Ada reference manual*. Heidelberg, Springer 1980, Lecture Notes in Computer Science 106

[4] Iglewski M., Madey J., Matwin S.: *Pascal. Język wzorcowy. Pascal 6000*. WNT Warszawa 1979

[5] Knuth D., E.: Structured Programming with Gotos. W *Computing Surveys 6*, 4 December 1974.

JAN WALASEK

System autorski ASYS 3.0

dokończenie ze s. 19

Planowana jest również rozbudowa i modernizacja edytora graficznego oraz modelu zapisu obiektów systemowych.

* * *

W trakcie poszukiwania odpowiednich narzędzi informatycznych użytkownik ocenia ich przydatność z perspektywy konkretnych zastosowań. Z tego względu projektant oprogramowania narzędziowego winien mieć również na uwadze potrzeby odbiorcy. Omówiony w artykule pakiet autorski był projektowany z przeznaczeniem dla nauczyciela szkoły średniej ogólnokształcącej. Przed przystąpieniem do realizacji projektu dokonano oceny stopnia złożoności obiektów symulacji dla wybranych działów: matematyki, fizyki i techniki. Wybór działu, o którym mowa wyżej, miał na względzie tzw. podatność graficzną (symulacyjną) problemu.

Zdaniem autorów artykułu opisany system ASYS pokrywa znaczną część znanych (z opracowań profesjonalnych) narzędzi niezbędnych do produkcji materiałów CAI z prostą warstwą interakcji. Nie bez znaczenia jest również niska cena wyposażenia sprzętowego i programowego, umożliwiającego efektywną pracę z proponowanym edytorem.

LITERATURA

[1] Gaik D., Gościniak I., Czapnik J., Piecha J.: System autorski ASYS – funkcje i zasady wykorzystania. IKZ Warszawa 1990, s. 30–40

[2] Piecha J.: The multilevel model for microcomputer ICAI systems. The International Journal of Applied Engineering Education. Vol. 5, No. 3'1989, Pergammon Press, Oxford 1989

[3] Piecha J.: Podstawy projektowania systemów CAI. INFORMATYKA 1991, nr 4, s. 1–6.

[4] Piecha J.: Proc. of Int. Conference on Computer Aided Learning and Simulation Technologies. Unesco Centre, Prague, June 1991.

Informatyka bankowa

W dniach 18 i 19 marca br. w salach konferencyjnych hotelu Mariott w Warszawie odbyło się saminarium „Informatyka bankowa i środki płatnicze". Organizatorami Seminarium byli: francuska agencja do spraw współpracy technicznej, przemysłowej i ekonomicznej ACTIM (*Agence pour la Coopération Technique, Industrielle et Economique*), a właściwie jej warszawska placówka – Francusko-Polski Ośrodek Informacji i Promocji Inwestycji CEPI (*Centre Franco-Polonais d'Information et de Promotion*), oraz Wydział Ekonomiczny Ambasady Francji w Polsce.

Bardzo obszerny program Seminarium (ponad 8 godzin dziennie) obejmował prezentację obecnej organizacji oraz informatyzacji i automatyzacji francuskiego systemu bankowego. Przedstawiciele 16 czołowych francuskich banków oraz instytucji związanych z informatyzacją i automatyzacją operacji bankowych wygłosili łącznie 19 referatów. Seminarium było adresowane do kierowniczej kadry, zwłaszcza komórek informatycznych, polskich banków, których kilkudziesięciu przedstawicieli wzięło udział w tej imprezie. O wadze, jaką strona francuska przywiązywała do Seminarium, świadczył osobisty udział ambasadora Francji, p. Alaina Bry, który dokonał oficjalnego otwarcia imprezy, a także udział jako referentów przedstawicieli centralnego banku Francji (*Banque de France*) oraz francuskiego ministerstwa gospodarki, finansów i budżetu (*Ministere de l'Economie, des Finances et du Budget*).

Oprócz referatów na temat organizacji francuskiego systemu bankowego oraz obiegu pieniądza, z punktu widzenia technologii informatycznej najbardziej interesującymi były następujące referaty:

„ELAN – system informatyczny banku Credit Lyonnais";

„Organizacja wewnętrzna banku", omawiający system informatyczny ATLAS II w Banque Nationale de Paris;

„System informatyczny banku Crédit Agricole – projekt GALAXIE";

„Integrator systemów bankowych", omawiający zaadaptowany do specyfiki polskich banków szeroko stosowany we Francji kompleksowy, modułowy system informatyczny NOVA–BANK;

„Informatyzacja polskiego banku", omawiający doświadczenia firmy SLIGOS przy wprowadzeniu w Polsce systemu PABA;

„Oferta zintegrowanego oprogramowania dla polskiej bankowości", zawierający propozycję sprawdzonych rozwiązań firmy STERIA BANQUES;

„Bankowy system informacyjny i produkcyjny", zawierający propozycję rozwiązań firmy BULL;

„Od strategii do rozpowszechnienia informatyki: szkolenie polskich bankowców", zawierający ofertę francuskiego centralnego ośrodka kształcenia bankowców CFPB (Centre de Formation de la Profession Bancaire);

„Środki płatnicze i systemy rozliczeń międzybankowych we Francji";

„System rozliczeń w banku Crédit Agricole – przykład CEDICAM;

„Automatyczne przetwarzanie czeków – urządzenia bankowe i handlowe";

„Karty kredytowe we francuskim systemie płatniczym";

„System informacji w elektronicznym obrocie pieniężnym";

„Automaty bankowe i samoobsługa bankowa";

„Przetwarzanie upoważnień oraz sieci transmisji".

Kompleksowość tematyki Seminarium pozwoliła polskim bankowcom poznać w znacznych szczegółach światowy poziom najnowszych rozwiązań informatycznych, zapewniających sprawne działanie gigantycznego obiegu pieniądza w rozwiniętej gospodarce. Do efektywnej percepcji treści Seminarium przyczyniła się wzorowa organizacja imprezy, zarówno bieżące tłumaczenia wygłaszanych referatów i starannie przygotowane materiały drukowane, jak i doskonała obsługa gastronomiczna, która pozwoliła uczestnikom kondycyjnie wytrzymać wyjątkową intensywność tak obszernego programu imprezy.

/mail

UNIX nad Zatoką San Francisco

W samym środku kalendarzowej zimy, jak zwykle tutaj łagodnej – rano trochę szronu, w południe za ciepło w swetrze – San Francisco stało się na kilka dni niekwestionowanym światowym centrum UNIX-a. W ciągu jednego tygodnia – między 20 a 24 stycznia – spotkały się tu dwie cykliczne imprezy UNIX-owe najwyższej rangi: zimowa konferencja stowarzyszenia USENIX i międzynarodowa wystawa UNIX-owa organizowana przez UniForum. Atrakcyjność miejsca i bliskie sąsiedztwo Doliny Krzemowej sprawiły, że w obu przypadkach były to imprezy typu „wszyscy byli" – korytarze konferencyjnego Hiltona aż roiły się od takich znakomitości, jak Dennis Ritchie, Kirk McKusick czy Peter Honeyman, a na wystawie nie zabrakło bodaj żadnej liczącej się firmy. Szczęśliwy splot okoliczności sprawił, że i ja tam byłem, czego wynikiem niniejszy raport specjalny dla czytelników INFORMATYKI.

KONFERENCJA

Najpierw kilka słów o organizatorze. Powstałe w 1975 roku **USENIX Association** jest stowarzyszeniem profesjonalistów – programistów systemowych i użytkowych, administratorów systemów, naukowców i wykładowców – zawodowo związanych z UNIX-em bądź z UNIX-opochodnymi systemami operacyjnymi. Liczy kilkanaście tysięcy członków, głównie z USA (jestem bodajże jedynym członkiem polskim) i jest organizacją prężną i dość wpływową. USENIX próbuje stymulować nieskrępowany rozwój UNIX-a, stara się przeciwdziałać zagrożeniom związanym z przedwczesną bądź nieodpowiedzialną standaryzacją (m.in. poprzez swoje „wtyczki" w standaryzacyjnych grupach roboczych), jest jedynym w USA liczącym się lobby UNIX-owych fachowców. Obszar zainteresowań stowarzyszenia jest wyraźnie otwarty w przyszłość – wprawdzie typowy USENIX-owiec o OS/2 wyraża się z niesmakiem, ale bardzo interesują go systemy takie, jak Mach czy Chorus.

USENIX wydaje dwa periodyki: ukazujący się co dwa miesiące wewnętrzny biuletyn; ***login***: oraz wysoko notowany kwartalnik naukowo-techniczny ***Computing Systems***. Natomiast praktycznie jedynym (jeśli nie liczyć Usenet-owych grup dyskusyjnych) miejscem spotkań członków są konferencje, a przede wszystkim dwie główne, odbywające się rokrocznie – zimowa i letnia. (USENIX organizuje co roku kilka do kilkunastu konferencji, seminariów i spotkań roboczych). Uczestnictwo w konferencjach jest dosyć kosztowne – dotyczy to zwłaszcza wykładów monograficznch, o których za chwilę – na szczęście płacą zwykle pracodawcy.

Program konferencji Zima 1992 był tradycyjnie już przeładowany (zmora większości konferencji – Hilton kosztuje, urlopy uczestników również) i niestety nie sposób było być wszędzie tam, gdzie by się chciało. Dotyczyło to w równej mierze wykładów monograficznych (pierwsze dwa dni konferencji), wieczornych spotkań w grupach zainteresowań, jak i samych sesji technicznych (trzy ostatnie dni). Moje sprawozdanie jest więc z konieczności selektywne i odzwierciedla prywatne zainteresowania – za co z góry czytelników przepraszam.

Wykłady monograficzne

USENIX-owe konferencje zdecydowanie nie są imprezami dla początkujących – z wyjątkiem właśnie wykładów monograficznych. Sesje takich wykładów, wprowadzone kilka lat temu, ze zrozumiałych względów cieszą się dużym powodzeniem – jest to najszybszy (i mimo wszysto najtańszy) sposób zdobycia dużej dawki rzetelnej wiedzy, zwykle od najlepszych specjalistów. Tym razem w tym swego rodzaju prologu konferencji wzięło udział kilkaset osób (z prawie półtora tysiąca uczestników konferencji), w sumie odbyło się dziewiętnaście sesji jednodniowych (9–17, z przerwą na lunch) i jedna dwudniowa.

Rozrzut tematyczny owych sesji szkoleniowych był dość duży – od administrowania systemem dla początkujących i zaawasowanych (w tym także o AFS i DCE/DFS), zagdnień bezpieczeństwa (Kerberos!) i szeroko rozumianej problematyki sieciowej, poprzez minikursy C, C++ i Perl, aż do specjalistycznych wykładów o budowie UNIX-owego jądra. Nie bez powodu właśnie wykłady o wnętrzu UNIX-a (trzy sesje: o OSF/1, 4.4BSD i o SVR4, ta ostatnia dwudniowa) wzbudzały największe zainteresowanie – aktualne publikacje na ten temat właściwie nie istnieją.

Jedynej wady instytucji wykładów monograficznych – ich współbieżności – raczej nie da się uniknąć. Na szczęście zawsze można przyjechać na następną konferencję, większość wykładowców powtarza swoje wykłady wielokrotnie.

Sesje techniczne

Po dwudniowej sztafecie szkoleniowej rozpoczęła się właściwa konferencja, czyli sesje techniczne. Tu współbieżność była już mniejsza – w tym samym czasie odbywały się referaty i wykłady zaproszonych gości – ale trudności z wyborem mieli wszyscy. Zwykle w takich przypadkach wbierano wykład, jako że grubą księgę materiałów konferencyjnych z tekstami referatów uczestnicy otrzymywali już pierwszego dnia konferencji. Nie zawsze wybierano słusznie.

Sesje techniczne miały kilka punktów kulminacyjnych. Dość nieoczekiwanie pierwszym, od razu na wstępie okazał się odczyt otwierający **Mitchella Kapora**, założyciela i wieloletniego szefa Lotusa, a obecnie prezydenta Electronic Frontier

Foundation. Audytorium USENIX-owych konferencji na ogół niechętnie słucha prelegentów z tego typu tytułami, lecz Kapor mówił rozsądnie i ciekawie o pilnej potrzebie zajęcia się tworzeniem **ogólnokrajowej publicznej sieci informatycznej,** zanim politykom, businessmanom i innym ignorantom uda się zrobić z niej wielkie śmietnisko w rodzaju komercyjnej telewizji. Należy tu korzystać z doświadczeń Internetu, który jakkolwiek z założenia nie jest siecią publiczną, to jednak ucieleśnia szereg fundamentalnych zasad o znaczeniu ogólnospołecznym. Kapor z wyraźną zazdrością odnosił się do francuskich sukcesów z ISDN.

Z zaproszonych wykładowców bodaj największe audytorium zgromadził prof. **JOHN Ousterhout** z University of California, Berkeley. Jego wykład – w moim prywatnym rankingu najlepszy – przedstawiał dwa opracowane przezeń pakiety **TcL** (czytaj: tykl) i **Tk**. Tcl to prosty, uniwersalny, rozszerzalny język poleceń przeznaczony do zanurzania w aplikacjach. Tk rozszerza zakres stosowania Tcl na środowisko X Window System – funkcjonalnie Tk udostępnia skryptom języka Tcl możliwości podobne do realizowanych przez pakiet Xt. Tcl, Tk i sporo związanego z nimi oprogramowania można znaleźć w rozmaitych publicznych archiwach Internetu. (Czynimy starania o możliwość druku w INFORMATYCE tłumaczeń dwóch źródłowych artykułów o Tcl i Tk).

Dla zainteresowanych **standaryzacją UNIX-a** przygotowano dwa wykłady. W pierwszym **Jeff Haemer** nakreślił mocno pesymistyczny obraz działań w tej dziedzinie rozmaitych amerykańskich i międzynarodowych komitetów i grup roboczych. Dowodził, że zły standard jest daleko gorszy niż żaden. W konkluzji namawiał słuchaczy do stworzenia środowiskowego lobby, które byłoby zdolne sutecznie wpływać na poczynania ciał standaryzujących. W drugim wykładzie **Martin Kirk**, z X/Open, zajął się bodaj najbardziej kontrowersyjnym obszarem prac normalizacyjnych – standaryzacją funkcji i procedur administrowania systemem. Przedstawił całkowicie oryginalny, obiektowy model administrowania systemem, który tworzy jedna z POSIX-owych grup roboczych. Audytorium miało na temat tego modelu uczucia mocno mieszane,– niektórzy chwalili jego spójność i klarowność, lecz większości wyraźnie nie podobała się perspektywa przekreślenia jednym ruchem całej UNIX-owej tradycji administrowania systemem. Należałem raczej do tych pierwszych, jakkolwiek w prowadzeniu prac stricte badawczych przez ciała standaryzacyjne rzeczywiście jest coś nienormalnego.

Dość interesujące były również dwa wykłady przedstawicieli przemysłu. **Fred Horman** z UNIX System Laboratories opowiadał o najnowszej wersji środowiska graficznego **OPEN LOOK,** wyraźnie dryfującego w stronę mariażu z bardziej popularnym Motifem – główna nowość to **MooLIT**, czyli Motif/OPEN LOOK Intrinsics Toolkit, który ma zastąpić dotychczasowy OLIT. Natomiast **Jennifer Steiner** i **Richard Mackey**, z Open Software Foundation, dokonali przeglądu **Distributed Computing Environment** (DCE). Dowiedzieliśmy się m.in., że wszystkie systemy korzystające z DCE będą widoczne poprzez globalną przestrzeń nazw z jednym wspólnym korzeniem oznaczanym jako "/..." (trzy kropki!).

Pora na kilka zdań (no, kilkadziesiąt) o referatach. Było ich w sumie ponad trzydzieści. Niestety – lub na szczęście, jak kto woli – nie sposób było wysłuchać wszystkich, ze względu na współbieżność z wykładami i wystawą, więc poniżej tylko o kilku moim zdaniem najciekwawszych. Tradycyjnie referenci mieli za mało czasu (20 min. referat, 10 min. pytania) i bili rekordy szybkości mówienia. Na szczęście nawet z ostatnich rzędów widać było twarz referenta – na ogromnym ekranie! W sumie przypominało to trochę stadion, tyle że atmosfera była robocza. Ale do rzeczy.

Zaletami (i wadami) ciągle niedocenianego mechanizmu **odwzorowywania plików w pamięci** zajęli się w swoim referacie **Orran Krieger**, **Michael Stumm** i **Ron Urnau**. Przypomnijmy, że jest to mechanizm, który pozwala zastępować operacje odczytu – zapisu dotyczące plików dyskowych (lub dokładniej – plików przechowywanych w blokowych urządzeniach o dostępie swobodnym) operacjami odczytu – zapisu dotyczącymi wirtulnej przestrzeni adresowej procesu. Unika się w ten sposób narzutów związanych z przepisywaniem danych pomiędzy buforami systemowymi a buforami w przestrzeni procesu – normalnie jest to niezbędne dla wszelkich plikowych operacji wejścia–wyjścia, tak „buforowanych" (biblioteka *stdio*) jak i „niebuforowanych" (systemowe funkcje we–wy). Działanie na odwzorowywanych plikach wymaga zwykle mniej wywołań funkcji systemowych, ale tu zysk należy pomniejszyć o zwiększony narzut związany z koniecznością obsługi większej ilości błędów stron. W sumie te i inne korzyści efektywnościowe są znacze, natomiast niewątpliwą wadą tego mechanizmu jest to, że operacje dotyczące odwzorowywanych plików z konieczności różnią się składniowo i semantycznie od klasycznych UNIX-owych operacji na wszystkich plikach w ogóle, co jest zarówno niewygodne, jak i nieeleganckie. Autorzy referatu rozwiązali (lub ściślej – prawie rozwiązali) ten problem projektując bibiotekę, która stanowi warstwę pośrednią pomiędzy odwzorowywanymi plikami a programami użytkowymi – przezroczystą dla programów „klasycznych" (nowa implementacja biblioteki stdio, emulacja systemowych funkcji we-wy), zaś tym „nieklasycznym" udostępniającą wygodniejszy zestaw funkcji do bezpośredniego operowania na odwzorowywanych plikach.

Alan Emtage i **Peter Deutsch** opowiadali o ***archie***, który jest rodzajem automatycznego rejestratora i skorowidza plików i pakietów publicznie dostępnych w Internecie. Ci wszyscy, którzy mieli okazję pobuszować trochę po Internecie, wiedzą doskonale jak przepastne są jego archiwa. Dostęp do nich nie nastręcza żadnych trudności (jeżeli przepustowość linii pozwala), wystarczy elementarna znajomość *ftp*, o ile tylko potrafimy zlokalizować plik lub pakiet, którego potrzebujemy (węzłów utrzymujących publicznie dostępne archiwa jest w Internecie grubo ponad tysiąc, w sumie przechowują około dwóch milionów plików). Do tego właśnie służy *archie*, eksploatowany od ponad roku i znany jak Internet szeroki – również w Polsce. Jest niewątpliwie tylko pierwszym krokiem na długiej drodze do budowy uniwersalnego sieciowego systemu lokalizacji zasobów, lecz jest również pouczającym przykładem skutecznego rozwiązania niełatwego problemu skromnymi środkami. (Więcej o *arche* spróbujemy napisać w INFORMATYCE niebawem).

Michael Kazar, z Transarc Corporation, przedstawił nowy system plików zaprojektowany dla wspomnianego już wcześniej DCE. System nosi nazwę **Episode** i ma zastąpić w roli lokalnego systemu plików dominujący obecnie tak zwany *szybki system plików* z 4.2BSD UNIX-a. Zaletą Episode będzie niewątpliwie dużo większa niezawodność – system działa trochę jak baza danych rejestrując historię swoich modyfikacji (trzeba

jednak dodać, że niektóre decyzje implementacyjne wzbudziły u słuchaczy sporo wątpliwości). A poza tym oczywiście pełna zgodność z wszystkim, co było do tej pory – na przykład Episode ma gładko współpracować z NFS.

Bill Cheswick, administrator sieci z AT&T Bell Laboratories, skądinąd znany *hacker*, opowiadał **o zabawie w kotka i myszkę ze zdalnym włamywaczem**, któremu wydawało się, że już jest w środku – to znaczy już ma uprawnienia superużytkownika – gdy tymczasem działał w specjalnie dla niego stworzonym sztucznym środowisku systemowym (*chroot* plus inne sztuczki). Referat był mocno techniczny, a przy tym niezwykle zabawny, choć temat raczej ponury – w ciągu trzech lat od niesławnego wyczynu Morrisa (internetowy „robal" z 1988 roku) liczba incydentów tego typu zwiększyła się kilkakrotnie, a prawo w wielu krajach jest równie nieprzygotowane jak wtedy – główny bohater referatu został wprawdzie zlokalizowany – w Holandii – lecz w świetle tamtejszego prawa nie popełnił przestępstwa.

Sandeep Khanna, Michael Sebree i **John Zolnowsky** z SunSoft (faktycznie dział Sun Microsystems) przedstawili referat, który prawdopodobnie będzie wielokrotnie cytowany — mówili **o szeregowaniu** procesów (i wątków) **w SunOS 5.0.** Jak wiadomo, klasyczny UNIX, system stricte wielodostępny, zupełnie nie nadawał się do zastosowań wymagających przetwrzania w czasie rzeczywistym. We wszystkich wersjach poprzedzających SVR4 procesy wykonujące kod jądra były chronione przed wywłaszczeniem, a użytkownicy nie mieli praktycznie żadnej możliwości wpływania na priorytety procesów. Pierwsze zmiany w tym zakresie przyniósł SVR4, ale wygląda na to, że przełomu dokona tu dopiero spóźniający się wciąż SunOS 5.0 (Sun-owa implementacja-nadzbiór SVR4, główny element Solaris 2.0) Tego akurat referatu nie podejmuję się streszczać w kilku zdaniach, dla zainteresowanych tylko jedna informacja – różnica między czystym SVR4 a SunOS 5.0 polega między innymi na tym, że o ile ten pierwszy implementuje w kodzie jądra **punkty**, w których możliwe jest wywłaszczanie, o tyle ten drugi dopuszcza wywłaszczanie **wszędzie** poza nielicznymi, krótkimi fragmentami kodu jądra. Wydaje się, że nawet ta pojedyncza informacja nieźle tłumaczy, dlaczego firma Sun, współtwórca SVR4, tak długo pracuje nad swoją impementacją tego systemu. Wygląda na to, że jest na co czekać.

Interesujący okazał się również referat **Bruce Nelsona** i **Yu-Ping Chenga** z Auspex Systems **o wyższości dysków SCSI nad dyskami IPI** w komputerach, których głównym zadaniem jest udostępnianie plików poprzez **NFS**. Sprzęg SCSI jest, jak wiadomo, generalnie inteligentniejszy niż sprzęg IPI, a w dodatku jego inteligencja jest rozproszona – każdy dysk ma własny sterownik, dzięki czemu zwiększanie liczby dysków (potencjalnie do 15 na jednej szynie SCSI-2) w warunkach pracy z NFS niemal liniowo zwiększa przepustowość. Jak się okazuje, w przypadku IPI przepustowość w analogicznych warunkach szybko się nasyca. Powód jest prosty i wynika bezpośrednio ze specyfikacji NFS-u (a ściślej z definicji używanego przez NFS protokołu UDP): dane przesyłane są w blokach nie dłuższych 8K. Wielodyskowe systemy IPI, szybsze w sekwencyjnym dostępie do dużych plików, w przypadku plików NFS-owych przegrywają z teoretycznie wolniejszymi SCSI, nie dorównując tym ostatnim zdolnością zrównoleglania operacji. W ten sposób referat potwierdził to, co niektórzy administratorzy sieci lokalnych podejrzewali od dawna.

Zupenie nietypowy był referat **Sharon Hopkins** – tematem była **poezja w języku Perl**. Perl jest nowym, niezwykle modnym językiem programowania, swego rodzaju połączeniem C, sh, awk i jeszcze pewnie czegoś. Jego twórcą jest Larry Wall, bardzo barwna postać (również dosłownie, ale o tym za chwilę). Perl ma tak zwolenników, jak i zdecydowanych przeciwników – ktoś powiedział, że to „awk z rakiem skóry". Język chyba rzeczywiście nie jest piękny, lecz niewątpliwie jest wyrazisty – dowodem tego całkiem niezłe wiersze, które czytała Sharon. Niektóre z nich dają się z sensem wykonywać – i produkują kolejne wiersze!

Na zakończenie **Alan Kaplan**, z AT&T Bell Laboratories, nie bez nostalgii opowiadał o niemal nieznanym, w całości asemblerowym, specjalizowanym wariancie UNIX-a, który pobił swojego rodzaju rekord długowieczności – równo dwadzieścia lat nieprzewanej, bezawaryjnej eksploatacji w kilkuset egzemplarzach. System nazywał się **COSNIX**, powstał w 1971 roku jako platforma dla systemu przetwarzania transacji przeznaczonego dla lokalnych oddziałów Bella, a jego bezpośrednim przodkiem był równie nieznany, też asemblerowy **UNIX-A**. Kończąc Kaplan podzielił się ze słuchaczami taką oto pozornie tylko banalną refleksją: jak wielką moc obliczeniową mogłyby mieć dzisiejsze komputery, gdyby oprogramowanie systemowe było równie proste i sprawne jak kiedyś.

Nocne rozmowy

Każdego wieczoru po zakończeniu sesji technicznych rozpoczynały się liczne nieformalne spotkania seminaryjno-dyskusyjne, czyli sesje **BOF** (*Birds-Of-a-Feather*, z przysłowia, dalej jest *flock together*). Tu znów współbieżność uniemożliwiała bycie wszędzie, gdzie by się chciało – a szkoda, gdyż spotkania BOF są bodaj najlepszym miejscem wymiany doświadczeń i plotek, a także swego rodzaju targowiskiem niepublikowanych i półpoufnych informacji.

Mnie udało się posłuchać o przyszłości UNIX-owego wieloprzetwarzania, a ściślej o decyzjach implementacyjnych twórców **SVR4.2 ES/MP** (ciekawostka: wygląda na to, że RFS umrze śmiercią naturalną – nikt nie chce podjąć się jego paralelizacji), porozmawiać z członkami **CERT** (*Computer Emergency Response Team*) o organizacyjno-systemowych aspektach bezpieczeństwa UNIX-owych sieci i wziąć udział w spotkaniu informacyjno-dyskusyjnym na temat przyszłości **UNIX-a z Berkeley** (4.4BSD ma być gotowy wiosną 1993, wcześniejszą wersję uwolniła całkowicie od kodu AT&T firma BSDI Roba Kolstada – system do kupienia wraz z kodem źródłowym, tanio!).

Koloryt lokalny

Kilka słów o uczestnikach i atmosferze. USENIX-owcy tworzyli barwną i socjologicznie ciekawą grupę społeczną. Zdecydowanie przeważali mężczyźni nieco po trzydziestce, najczęściej biali, rzadziej pochodzenia azjatyckiego, stosunkowo niewiele kobiet. Dominowały swetry i podkoszulki, nie zauważyłem ani jednego garnituru. Wielu krótkowidzów. Barwnością stroju wyróżniał się wspomniany już Larry Wall – malinowa marynarka, jaskrawozielona koszula z żabotem, muszka, itd. Zupełnie inaczej prezentowali się zaglądający na konferencję uczestnicy wyraźnie marketingowej imprezy UniForum, o której dokładnie za chwilę. Tam obowiązywał mundur (oni to nazywają *dressing code*): ciemny garnitur, jedwabny krawat i prosto od fryzjera. Larry'ego by tam chyba nie wpuścili.

Uczestnicy tak tej, jak i innych USENIX-owych konferencji to z reguły wybitni profesjonaliści, często znani *hackerzy* (nie mylić z *crackerami*, *hacker* to postać pozytywna, *crackerów* nikt nie szanuje). Prawie wszyscy eksperymentowali kiedyś

z kodem jądra, czy to na studiach, czy w pracy. Mniej więcej połowa to administratorzy dużych systemów sieciowych, reszta to programiści systemowi, naukowcy i studenci. Panuje kult fachowości i wyraźna niechęć do speców od marketingu. Za najlepszy system operacyjny świata uważa się oczywiście UNIX, przede wszystkim BSD, SunOS, a ostatnio również SVR4. AIX nie ma zbyt wielu zwolenników, głównie ze względu na sporo niezgodności z wersjami uznawanymi za wzorcowe. Z innych systemów operacyjnych szanuje się jeszcze tylko VMS. Określenia „systemy otwarte" nie używa się wcale, uchodzi za pusty zwrot marketingowy.

Poza samym UNIX-em największym szacunkiem USENIX-owcy darzą Internet i rozliczne związane z nim instytucje i ułatwienia. Niektórzy znają się od lat wyłącznie poprzez Sieć, są i tacy, którzy z góry wykluczą możliwość pracy nie gwarantującej dostępu do Internetu. Wszyscy szanują protokoły TCP/IP, model ISO/OSI tylko nielicznie.

A poza tym to bardzo mili ludzie.

Inne atrakcje

Wspomiałym pomysłem okazało się wyprowadzenie oficjalnego konferencyjnego przyjęcia z Hiltona do słynnego **Exploratorium** (coś w rodzaju muzeum nauki i techniki z założeniem aktywnego udziału zwiedzających, wszystkie eksponaty służą do doświadczeń). W domu byłem po północy.

Podczas konferencji ukazywała się gazeta codzienna o tytule – jakże by inaczej – **/etc/motd.** Zawierała głównie informacje, w tym również o rozmaitych atrakcjach San Francisco, niektóre nie do znalezienia w przewodnikach. Redagował Peter Salus, z SUN User Group.

WYSTAWA

UniForum, organizator wystawy, to stowarzyszenie o charakterze zdecydowanie innym niż USENIX. Zrzesza przede wszystkim poważniejszych użytkowników systemów UNIX-owych, programistów użytkowych oraz – co ważne – fachowców od marketingu i członków zarządów firm i korporacji o interesach związanych z UNIX-em. Wydaje miesięcznik *UniForum Monthly* (na kredowym papierze, USENIX przeciwnie – publikuje najtaniej jak można) i szereg półregularnych biuletynów. Podobno ma polskich członków.

W ciągu trzech dni (22–24 stycznia, a więc równolegle z USENIX-owymi sesjami technicznymi) prawie trzystu wystawców prezentowało niezwykle szeroki wachlarz produktów sprzętowych i programowych, zresztą nie tylko – były również wydawnictwa. Inaczej niż Comdex czy CeBit, impreza UniForum jest wystawą wyłącznie UNIX-ową, a więc żadnych dzieci, hobbystów i programów zwalczających wirusy. Co za ulga!

Jak się rzekło, byli wszyscy. IBM i HP, Sun i DEC, Cray i Amdahl, Fujitsu i Bull, WordPerfect i Frame, Sybase, Oracle i Informix, UNIX System Laboratories i Open Software Foundation, X/Open i 88open Consortium – by wymienić tylko niektórych. Nawet o samych nowościach nie da się opowiedzieć w kilku zdaniach. Wybór musiałby prowadzić do posądzeń o ukrytą reklamę, więc spróbuję zająć się raczej trendami niż produktami. (Na marginesie cytat: „UNIX dzisiaj to nie konkretny produkt, to sprzęg" – Scott McNealy, szef Sun Microsystems.)

Najbardziej rzuca się w oczy **dominacja X Windows**. Niemal nie było stoiska bez X-terminala czy X-stacji roboczej. Mimo zastrzeżeń fachowców („w X nawet światło biegnie dwa razy wolniej") jest to bez wątpienia produkt roku. Rywalizacja Motif contra Open Look trwa, ale przewaga Motifu jest coraz wyraźniejsza.

Drugi przebój to szeroko rozumiane **przetwarzanie rozproszone**. Tu rywalizacja między DCE (lansowanym, przypomnijmy, przez OSF) a UI-ATLAS (UNIX International) dopiero się zaczyna i wszystko jest tu możliwe – nawet unifikacja. Z drugiej strony – Oracle już od dawna zapowiada w pełni rozproszoną wersję swego produktu, to może być przebój roku.

Jest też szereg innych niewygasłych rywalizacji. Niektóre dotyczą rozwiązań wewnętrznych systemu operacyjnego (jądro tradycyjne contra mikrojądro – patrz niżej, a jeśli mikrojądro, to Mach czy Chorus), inne protokołów sieciowych (coś jakby odwrót od ISO/OSI w stronę TCP/IP), jeszcze inne rodzaje procesora (CISC contra RISC, a jeśli RISC, to SPARC czy MIPS – na razie przeważa pogląd, że RISC, i raczej SPARC). Na szczęście – tak dla użytkowników, jak i dla programistów – rozdmuchana nieco rywalizacja OSF/1 i SVR4 zaczyna tracić na znaczeniu: **OSF/1 ma zostać doprowadzony do zgodności** z najnowszą edycją System V Interface Definition, co oczywiście oznacza zgodność (na poziomie sprzęgu) **z SVR4,** a nie, jak to miało być pierwotnie – z SVR3. Bardzo dobra wiadomość.

Styczniowa impreza UniForum była przede wszystkim wystawą – lecz nie wyłącznie Tak naprawdę równolegle odbywała się również konferencja, rozmachem podobna do USENIX-owej, lecz znacznie mniej techniczna – inne audytorium – i dla mnie osobiście mało ciekawa. Może jedynym wyjątkiem był panel na temat wyższości rozwiązań typu Mach czy Chorus (mikrojądro, ang. *microkernel*), nad tradycyjnymi (jądro monolityczne, ang. *monolithic kernel, monolith*) – lub odwrotnie. Obie spierające się strony (David Black, z OSF i Daniel Julin, z Carnegie-Mellon University contra Kirk McKusick, współtwórca 4.4BSD) miały swoje racje, osobiście chętnie bym posłuchał, co myśli o tym Dennis Ritchie, kiedyś twórca UNIX-a, a dziś *Planu 9.*

UNIX A SPRAWA POLSKA, CZYLI WNIOSKI I REFLEKSJE

Jakie wnioski z tego wszystkiego płyną dla nas, w większości wciąż początkujących UNIX-owców, w kraju bez pieniędzy? Myślę, że aby stworzyć słyszalne UNIX-owe lobby potrzebne są rzeczywiście silne UNIX-owe organizacje, najlepiej właśnie dwie – jedna profesjonalna, druga otwarta, angażująca również ludzi businessu. To już prawie mamy. Jednym z podstawowych celów obu powinno być przy tym doprowadzenie do świadomości decydentów dwóch również ważnych prawd – tego, że maszyny DOS-owe i ich bezpośredni następcy nie nadają się do poważnych ,profesjonalnych zastosowań, a także tego, że w kraju niemal bez zaszłości w dziedzinie systemów przetwarzania transakcji naprawdę nie ma powodu topić pieniędzy w przestarzałe rozwiązania typu systemów CICS-owych.

A po drugie, musimy wywalczyć sobie jak najpowszechniejszy, normalny dostęp do Internetu. Taki z realną możliwością korzystania z *ftp* (przepustowość!) i udziału w Usenet-owych grupach dyskusyjnych. Bez tego zawsze będziemy prowincją.

Moje uczestnictwo w Konferencji prawdopodobnie nie byłoby możliwe bez pomocy Ellie Young i Judy DesHarnais z władz USENIX Association. Jeszcze raz dziękuję.

JACEK SURMA

Berkeley

OS – 9
Siła i słabość standardu

dokończenie ze s. 16

pokazuje nie tylko, że pSOS$^+$ jest szybszy niż OS-9. Ujawnia także, że ten drugi nie jest w pełni deterministyczny w swych reakcjach. Przewagą pSOS$^+$ jest też możliwość pracy wieloprocesorowej;

- wykorzystanie bogatego środowiska do produkcji oprogramowania systemu UNIX – VxWorks [5] daje w tym względzie większe możliwości niż OS-9;
- zgodność z unormowaniami standaryzacyjnymi – POSIX (Portable Operating System UNIX [1] to standard interfejsu programowego, który ma gwarantować przenośność użytkowego kodu źródłowego. W jego ramach podgrupa robocza nazywana Posix 1003.4 przygotowuje dokument o nazwie „Rozszerzenia czasu rzeczywistego w przenośnych systemach operacyjnych". Firma Lynx Real-Time Systems (Campbell, USA) twierdzi [8], że jej produkt – LynxOS – jako jedyny zapewnia przenośność oprogramowania będąc zgodnym ze standardem POSIX 1003.4. Może to być prawdą tylko częściowo, w odniesieniu do wersji roboczych standardu, gdyż ostateczna edycja POSIX 1003.4 nie została jeszcze uchwalona. Co więcej, bezpośredni konkurent, firma VenturCom Inc., wysuwa oskarżenia, że w LynxOS brakuje niektórych wywołań systemowych zdefiniowanych przez AT&T. Z kolei jej system VENIX budzi wątpliwości [10], że co prawda jest on zgodny z UNIX-em w zakresie, jaki przewiduje licencja AT&T, lecz uczynienie go systemem czasu rzeczywistego wymagało wprowadzenia niestandardowych rozszerzeń.

Czy uchwalenie POSIX 1003.4 będzie oznaczać początek końca OS-9? Nowy system Microware – OS-9000 – jest argumentem za odpowiedzią przeczącą. Uzupełnia wskazane poprzednio atuty de facto standardu o możliwość przenoszenia oprogramowania między komputerami z mikroprocesorami 680 × 0, 80386, 80486 i 88000.

LITERATURA

[1 Eck Ch.: Real-time standards. Computer Design, No. 10, 1991
[2] MEN Mikro Elektronik GmbH: Katalog systemów VMEbus. 1991
[3] Microware Systems Corporation: OS-9 Catalog. 1991
[4] Microware Systems Corporation: OS-9 Technical Manual
[5] Microware Systems Corporation: Using Professional OS-9
[6] Nałęcki K.: Dlaczego UNIX nie jest systemem czasu rzeczywistego? INFORMATYKA 1991, nr 6, s. 2
[7] Oettle&Reichler Industrial Computers GmbH: The Industrial VMEbus'92
[8] Singh I.: Posix. Computer Desing, No. 3, 1991
[9] Trojniar W.: Metody implementacji systemów czasu rzeczywistego. INFORMATYKA, 1991, nr 6, s. 8
[10] Williams T.: Competitors claims kickoff real-time UNIX squabble. Computer Design, No. 9, 1991.

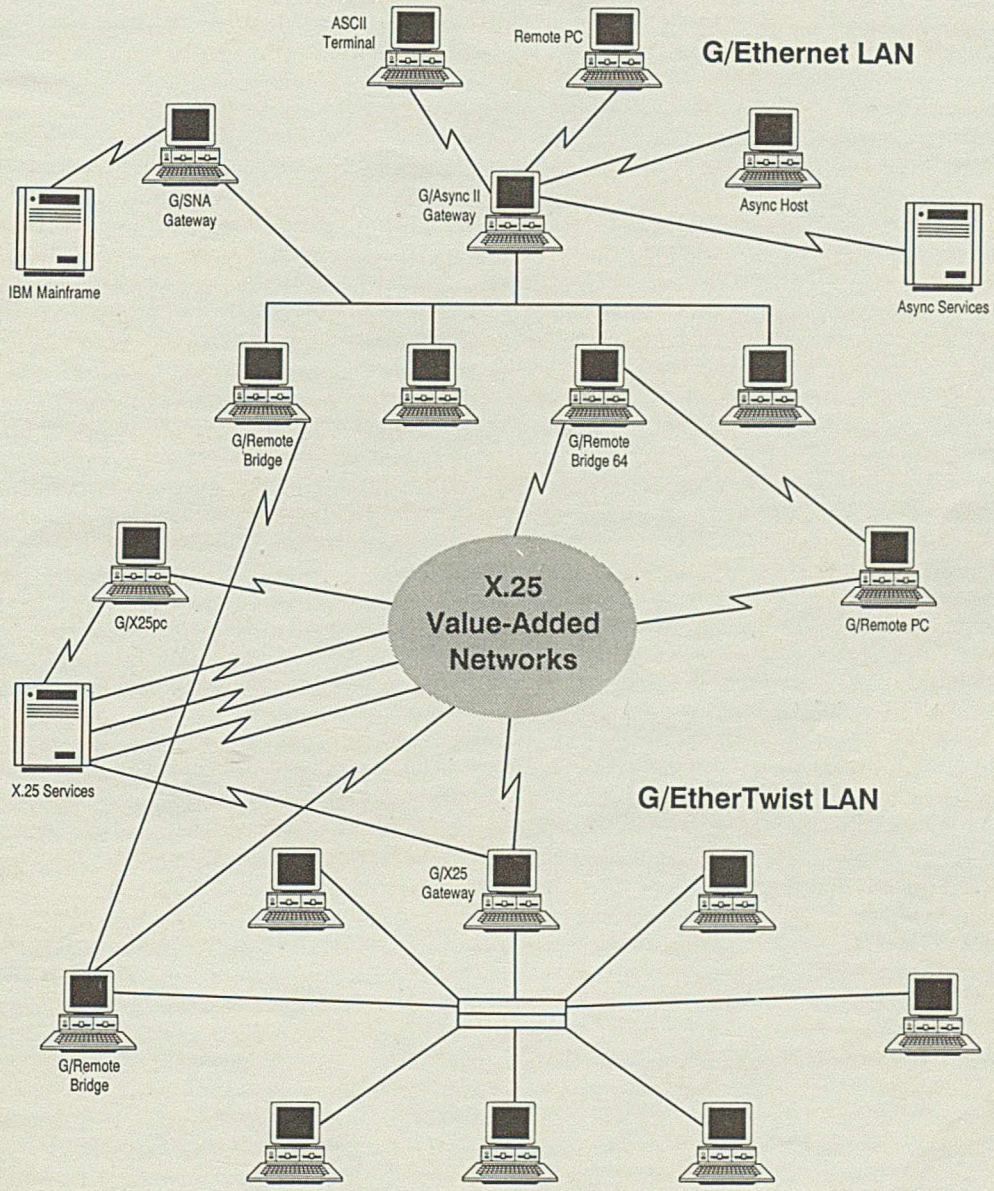
Rozstrzygnięcie konkursu PTI

10 grudnia ub.r. nastąpiło we Wrocławiu rozstrzygnięcie VIII Ogólnopolskiego Konkursu na najlepsze prace magisterskie z informatyki. Komisja konkursowa pod przewodnictwem doc. dr inż. *Czesława Danilowicza* przyznała następujące nagrody i wyróżnienia:
pierwszą nagrodę (3 mln zł) *Igorowi Walukiewiczowi* z Instytutu Informatyki Uniwersytetu Warszawskiego za pracę pt. **Gentzenowska aksjomatyzacja zdaniowej logiki PAL;**
drugą nagrodę (2 mln zł) *Ewie Kowalskiej* z Wydziału Informatyki i Zarządzania Politechniki Wrocławskiej za pracę pt. **Komputerowe wspomaganie nauczania o szyfrowaniu informacji;**
trzecią nagrodę (1,5 mln zł) *Dariuszowi Litwinience* oraz *Janowi Zatopińskiemu* z Instytutu Cybernetyki Technicznej Politechniki Wrocławskiej za pracę pt. **Język symulacji i symulator przepływu detali przez elastyczny system produkcyjny,**
trzy równorzędne wyróżnienia (po 1 mln zł): *Bożenie Bartoszek* z Instytutu Informatyki Politechniki Śląskiej za pracę pt. **Równoległa implementacja algorytmu dla znajdowania subminimalnej bazy cyklowej,** *Monice Salach-Gołysce* z Instytutu Informatyki Politechniki Warszawskiej za pracę pt. **System wspomagania generacji graficznych danych wektorowych na podstawie obrazów rastrowych** oraz *Michałowi Śmiałkowi* z Instytutu Informatyki Politechniki Warszawskiej za pracę pt. **System aproksymacji danych eksperymentalnych funkcjami analitycznymi.**

Kowalski P.: T³ – program przetwarzania wielojęzycznych tekstów naukowo-technicznych INFORMATYKA 1922, nr 5, s. 1 Charakterystyka obecnie stosowanych na świecie programów do przetwarzania trudnych tekstów technicznych oraz omówienie konstrukcji, możliwości funkcjonalnych i sposobu użytkowania programu T³, szeroko stosowanego do tego celu w Europie Zachodniej i USA.	**Kowalski P.: T³ – a program for processing of multilanguage scientific-technical texts** INFORMATYKA 1992, No. 5, p. 1 Characteristics of the to-day applied programs for processing of difficult technical texts and discusssion of structure, functional possibilities and operation methods of the T³ program, which is broadly used for this purpose in Western Europe and USA.	**Kowalski P.: T³ – ein Programm zur Verarbeitung von mehrsprachigen wissenschaftlich-technischen Texten** INFORMATYKA 1992, Nr. 5, S. 1 Eine Charakteristik von heute angewendeten Programmen zur Verarbeitung von komplizierten technischen Texten und eine Besprechung von Struktur, Funktionsmöglichkeiten und Anwendungsmethoden des T³ – Programmes, das in Westeuropa und USA sehr verbreitet ist.
Marzec B.: OS-9 – siła i słabość standardu INFORMATYKA 1992, nr 5, s. 13 Charakterystyka systemu operacyjnego czasu rzeczywistego OS-9 oraz ocena jego obecnej i przyszłej pozycji rynkowej jako standardu przemysłowego.	**Marzec B.: OS-9 – strenght and weakness of the standard** INFORMATYKA 1992, No. 5, p. 13 Characteristics of the OS-9 real time operating system and evaluation of its actual and future market position as industrial standard.	**Marzec B.: OS-9 – die Kraft und Schwäche des Standardes** INFORMATYKA 1992, Nr. 5, S. 13 Eine Charaktersitik des OS-9-Echtzeitbetriebssystems und eine Beurteilung seiner heutigen und künftigen Position als Industriestandard.
Gaik D., Kotula J., Piecha J.: System autorski ASYS 3.0 **INFORMATYKA 1992, nr 5, s. 17** Charakterystyka opracowanego w Zakładzie Elektroniki i Informatyki Uniwersytetu Śląskiego systemu ASYS 3.0, przeznaczonego do projektowania programów dydaktycznych klasy CAI.	**Gaik D., Kotula J., Piecha J.: The ASYS 3.0 authoring system** INFORMATYKA 1992, No. 5, p. 17 Characteristics of the in Electronics and Informatics Laboratory of the Silesian University elaborated ASYS 3.0 system, which is devoted for CAI type didactic programs design.	**Gaik D., Kotula J., Piecha J.: ASYS 3.0-Autorsystem** INFORMATYKA 1992, Nr. 5, S. 17 Eine Charakteristik von dem in Anstalt für Elektronik und Informatik der Schlesier Universität erarbeiteten ASYS 3.0-System, das zur Projektierung von CAI-Kategorie didaktischen Programmen bestimmt ist.

Produkty sieciowe LAN i WAN

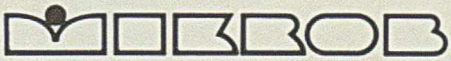
Gateway
communications, inc.



BEZKONKURENCYJNE PRODUKTY
SIECI KOMPUTEROWYCH NAGRADZANE PRZEZ:

PC MAGAZINE, LAN MAGAZINE, INFO WORLD

OFERUJE AUTORYZOWANY DYSTRYBUTOR:



P.W.P.T. MIKROB SP. Z O.O.

20-346 Lublin, ul. Długa 5
Tel. (0-81) 420-61, faks 415-43, teleks 643776 MIKRO PL

0/15/91

Na życzenie wysyłamy katalog produktów. Atrakcyjny program współpracy dla dealer'ów.

JUNISOFTEX Sp. z o.o.

44-100 GLIWICE ul. Konstytucji 11,
tel.-faks 31-75-10, 31-90-81 do 88 w. 250, 272, 282, teleks 036233

JUNISOFTEX – to firma z tradycjami i najdłuższymi doświadczeniami w eksploatacji wielodostępnych systemów komputerowych Novell w kraju.

JUNISOFTEX – to najstarszy w kraju wykonawca własnych wielodostępnych systemów pracujących w sieci NetWare firmy Novell.

JUNISOFTEX – to autoryzowany reseller amerykańskiej firmy Novell.

JUNISOFTEX – to dostawca zintegrowanych systemów komputerowych działających w kilkudziesięciu firmach obejmujących w każdym przedsiębiorstwie, niezależnie od formy własności i dziedziny gospodarki: finanse, majątek obrotowy, majątek trwały i nietrwały, sprzedaż, techniczne przygotowanie produkcji oraz kadry-płace.

JUNISOFTEX – to dostawca i wykonawca autoryzowanych sieci lokalnych firmy Novell.

JUNISOFTEX – to dostawca sprawdzonego sprzętu komputerowego renomowanych firm ALR, IBM, TEAM, DIGILAB, EPSON, MANNESMANN TALLY.

JUNISOFTEX – to wyłączny i autoryzowany dystrybutor komputerów i terminali firmy DIGILAB na Śląsku.

JUNISOFTEX – to dostawca i wykonawca okablowania lokalnych sieci komputerowych ARCNET, ETHERNET na kablach zwykłych i światłowodowych oraz na dwużyłowych kablach telefonicznych w przypadku stosowania oprogramowania NetWare ACCESS SERVER firmy Novell.

JUNISOFTEX – to nauczyciel, który chętnie podzieli się swoją wiedzą na organizowanych kursach w swojej szkole informatycznej – w tym nauczy Cię: podstaw informatyki, obsługi sprzętu komputerowego, eksploatacji własnych systemów informatycznych, edytorów tekstu: CHiWRITER, WordPerfect, arkuszy kalkulacyjnych QPRO v. 3.0, LOTUS 1-2-3 itp.

JUNISOFTEX – to doradca w zakresie komputerowych metod organizacji i eksploatacji systemów komputerowych.

JUNISOFTEX – to strażnik postępu i nowoczesności w Twojej firmie.

JUNISOFTEX – to gwarancja niezawodności, rzetelności i terminowości.

JUNISOFTEX – to Twój doradca i partner, któremu możesz zaufać, który Cię nigdy nie zawiedzie.

JUNISOFTEX – to partner, który Cię wysłucha i zawsze pomoże podjąć dobrą decyzję.

Jeżeli chcesz

- ☆ zastosować najnowocześniejsze systemy komputerowe oraz lokalne sieci komputerowe,
- ☆ zreorganizować, usprawnić i szybko skomputeryzować swoje przedsiębiorstwo,
- ☆ lepiej wykorzystać już istniejące w Twoim przedsiębiorstwie komputery,
- ☆ wymienić swoje niefortunnie zakupione oprogramowanie,
- ☆ pozbyć się problemów eksploatacyjnych, upadających systemów, nie działających komputerów,
- ☆ przeszkolić załogę swojego przedsiębiorstwa,
- ☆ dobrze zainwestować swoje pieniądze i podnieść rangę swojego przedsiębiorstwa,
- ☆ w przyszłości mieć dostęp do krajowej i światowej sieci komputerowej,

TO ZGŁOŚ SIĘ DO NAS!

0/17/91