

# INDEX

**NOTE** Page numbers followed by *n* refer to footnotes.

## Symbols

\$ (dollar sign), in AT&T assembly syntax, 9  
// comments, IDC command dialog and, 254  
% symbol, in AT&T assembly syntax, 9  
; (semicolon)  
    for comments, 108  
    for IDC statements, 252–253

## A

Abort command, 205  
accept\_file function, 349  
activation record, 67*n*. *See also* stacks, frames  
add\_auto\_stkpnt2 function, 378  
AddBpt function, 512, 556  
AddBptEx function, 512, 556  
AddCodeXref function, 556  
AddConstEx function, 556  
add\_dref function, 570  
add\_entry function, 353  
AddEntryPoint function, 556  
AddEnum function, 556  
AddHotkey function, 258, 556  
add\_seg function, 302, 303, 352  
add\_seg\_ex function, 302  
AddSourceFile function, 556  
add\_struct function, 301  
AddStructEx function, 556  
add\_struct\_member function, 301  
AddStructMember function, 556  
add\_til functions, 356  
add\_til2 function, 356  
add\_types function, 356–357

allins.hpp file, 235  
alset function, 294  
AltOp function, 556  
altvals, 291, 292  
AMD64 processor, IDA support for, 33  
Amini, Pedram, 204*n*, 485  
ana function, 371  
ana.cpp file, 371  
Analysis function, 556  
analysis.idc script, 196  
AnalyzeArea function, 556  
analyzer, in processor module, 366, 371–376  
anterior lines, for comments, 110  
anti-debugging, and x86emu plug-in, 454–455  
anti-dynamic analysis techniques, 433–437  
    debugger detection, 435–436  
    debugging prevention, 436–437  
    instrumentation detection, 435  
    virtualization detection, 433–435  
anti–reverse engineering, 418, 542  
anti–static analysis techniques, 418–432  
    disassembly desynchronization, 418–421  
    dynamically computed target addresses, 421–427  
    imported function obfuscation, 428–432  
    targeted attacks on analysis tools, 432  
API functions, signature information for, 228  
AppendFchunk function, 556  
application, for debugging process setup, 549  
ApplySig function, 556  
ar-style libraries, extracting information from, 232

- area control block, 304
- areach\_t data structure, 303
- area.hpp* file, 285, 303
- area\_t struct, 296
- arg\_ prefix for function parameters, 97
- arithmetic instructions, 11
- ARM processor, 48
- Array size dialog, 127
- arrays, 126–128
  - globally allocated, 132–133
  - heap-allocated, 135–136
  - IDC functions for manipulating, 256–257
  - iterating over contents for netnodes, 294–295
  - member access, 131–136
  - in netnode, 289
    - storing primary value within, 291
  - stack-allocated, 133–135
  - of structures, 141–142
- arrows window, in text view, 67
- ASCII string data window, 125
- ASCII strings, 69
  - determining reference to, 168
- ash global variable, 385
- AskAddr function, 556
- AskFile function, 260, 556
- askfile\_c function, 299
- AskIdent function, 556
- AskingUsingForm\_c function, 299–300
- AskLong function, 556
- AskSeg function, 556
- AskSelector function, 556
- AskStr function, 260, 556
- askstr function, 299
- AskUsingForm\_c function, 331
- AskYN function, 260, 556
- askyn\_c function, 299
- ASM files, 242–243
- asms array, 387
- asm\_t structure, 388
- ASPack, 426
- ASProtect, 426
- Assemble Instruction dialog, 239–241
- assembler, 4
  - assembly code, custom, 91
  - assembly languages, 4
    - creating mnemonic list for dis-assembly module, 367
    - emulating behavior, 274–277
- asynchronous interaction, with debugger, 519
- AT&T format, 8–9
- atoi function, 570
- atol function, 261, 570
- attaching debuggers, to running process, 498–499
- AttachProcess function, 556
- attributes of functions, editing, 117–120
- auto keyword (IDC), 252
- autocomments, 234
  - customizing, 112
- autogenerated names, 69, 104
  - for named location, 106
- auto.http* file, 285
- AutoMark function, 556
- AutoMark2 function, 556
- autonomous mode, for IDA, 196
- AutoShow function, 556
- AutoUnmark function, 556

**B**

- backdoor communication channels, in virtualization platforms, 434
- background color, of graph node, 185
- backward navigation, 85
- bad instruction <BAD> marks, 112–113
- base address
  - of array, 132
  - of IDA image, modifying, 47
  - specifying for program, 341
- base2file function, 354
- basic blocks, 63*n*, 176, 177
- Batch function, 556
- batch-processing mode, 187, 195–196
- BDS (Binary Diffing Suite), 467
- BeginEA function, 556
- beginner mode, for IDA, 206
- big-endian CPU, 10*n*
- bin* directory, for SDK, 281
- binary data file, 44
- binary diffing, 467
- Binary Diffing Suite (BDS), 467
- Binary File, in loader list, 47
- Binary File Descriptor library (libbfd), 24
- binary files, 4
  - comparing changes in revision, 466–467
  - vs. database files, 238
  - dynamic de-obfuscation, 522
  - loader for, 47–49
  - as only recognized file type, 337–338
  - release vs. debug, 412–414

- static de-obfuscation, 438–455
  - emulation-oriented
    - de-obfuscation, 443–455
    - scripted-oriented de-obfuscation, 438–443
  - stripping, 18
- binary image, reverse engineering, 339
- binary large object (blob), 289*n*
- binary reverse engineering, 417
- binary searches, 101–102
  - for byte sequence, 473
- BinDiff, 467
- BinNavi, 186, 484
- blacklist of users, 34
- blob (binary large object), 289*n*
- blocking operation, 280
- blocks of code, signatures for
  - identifying, 76
- Borland tools
  - compiler, IDA and, 403
  - and FILE pointer, 354
  - free command-line compiler, 410–411
  - Turbo Assembler (TASM), 9
- BOUNDS problem, 78
- BP-based frame, 119–120
- bpt\_NtContinue function, 543–544
- branching instructions, 11–12, 171
- breakpoint function, implementing, 535–536
- Breakpoint Settings dialog, 507
- breakpoints, 177
  - in emulator operation, 446
  - in IDA debugger, 499, 505–508
  - software, 437
- bss section, 72
- btree nodes, netnode content
  - storage in, 289
- buffer overflow, 270, 470
- Bug Scan appliance, 458
- bugs, reporting, 58
- BugScam, 463
- build environment, configuring, 283
- Burneye, 426
  - de-obfuscation, 448–453
  - program encryption with, 438–443
- byte code, 4
  - Python, 364–365
- Byte function, 259, 557
- byte offset, to field, 145
- byte\_patched notification code, 316
- bytes.hpp* file, 285, 385

- byteValue function, 570
- byte\_XXXXXX autogenerated names, 69

## C

- C
- calling convention, 87–89
  - in `nm` utility output, 21
  - parsing declarations, 150–151
  - parsing header files, 151–152
- C++
  - calling convention, 90
  - class definitions, 157–158
  - compilers
    - reverse engineering references, 165–166
    - type information embedded, 163, 404
  - libraries, 280
  - reversing primer, 156–166
- `c++filt` utility, 25
- call flow, 64*n*, 169, 171
- CALL instruction (x86), 86
  - emulator determination of target address, 450
- callback function, 518–519
- called functions, 86
  - recursive descent failure on return from, 12
- calling conventions, 86, 87–91
  - alternative for different compilers, 414–415
- `callui` function, 299
- `CanExceptionContinue` function, 557
- canonical feature flags, 367
- case sensitivity, in hex search, 102
- catalog of named constants, 114
- `cdecl` calling convention, 87
- `cdecl` functions, 118
- `_cdecl` modifier, 87
- `cfg` directory, 38, 201
- Change segment attributes dialog, 525
- character buffer, writing display text into, 380
- character-terminated strings, 124
- `charset` function, 294
- `charval` function, 294
- `charvals`, 291, 294
- `C_HEADER_PATH` option, in *ida.cfg* file, 203
- checkbox controls, on forms, 333
- choose function, 327
- choose2 function, 327, 329–330, 542
- ChooseFunction function, 557

- class connector function, 160*n*
- .class* file (Java), magic numbers to
  - identify, 16
- classification tools, 16–20
  - file command, 16–18
  - PE Tools, 18–19
  - PEiD, 19–20
- cleanup code, 161
- closing database files, 52–53
- cmd* variable, 371
- cmd.exe*, terminal, 189
- cmd.Operands* array, 373, 376
- CmtIndent* function, 557
- code
  - basic transformations, 110–122
    - code display options, 111–113
    - converting data to code, 121–122
    - formatting instruction operands, 114–115
    - manipulating functions, 115–121
  - cleanup, 161
  - custom assembly, 91
  - display options, 111–113
  - obfuscated. *See also* obfuscated code analysis
    - and compiler identification, 214
    - signatures for identifying blocks, 76
- code cross-references, 168, 169–171
  - IDA addition of, 462
  - IDC functions for, 264
  - SDK functions for, 303
- CODE XREF, 169
- collabREate, 488–491
  - capabilities by IDA version, 489
- collapsed node demo, 186
- collapsing blocks, 66
- collisions. *See* FLIRT
- color
  - assigning to node, 185
  - coding for names, 68
  - in IDA display, 207–208
  - in Linux console mode, 191
  - LST files with HTML tags, 245
  - in navigation band, 55
  - for output line portion, 380
- COMMAND function, 517
- command line, for IDA, 204
- comment.cmt* file, 235
- CommentEx function, 267, 557
- comments
  - customizing, 112
  - embedding in databases, 108–110
  - function, 110
  - IDC command dialog and, 254
  - IDC script for locating and tagging, 460
  - mangled names as, 162
  - for plug-ins, 311
  - predefined, with *loadint* utilities, 234–236
  - for processor modules, 382–383
  - for signature files, 212
  - for structure field, 145
  - in text view, 67
- Comments function, 557
- compact\_til* function, 358
- compilation, as lossy process, 5
- Compile function, 557
- compiler differences
  - alternative calling conventions, 414–415
  - debug vs. release binaries, 412–414
  - jump tables and switch statements, 400–404
  - main function location, 405–412
  - RTTI implementations, 404
- compilers, 4
  - determining which for building executable, 19
- identification
  - in IDA initial analysis, 51
  - and obfuscated code, 214
- startup sequences, 76
- validation, 7

- compound statements, in IDC
- scripts, 253
- compression
- of database component files, 52
- of obfuscated programs, 525
- computer licenses, 33
- concrete classes, 158
- conditional branching instructions, 11, 171
- conditional breakpoints, 506
- creating, 507
- on *NtContinue* function, 543
- conditional jumps, 64
- in text view, 67
- configuration files, 201–207
- exceptions.cfg* file, 539
- ida.cfg* file, 202–203
- idagui.cfg* file, 202, 203–206, 238, 251
- hotkey configuration, 204–205
- idatui.cfg* file, 202, 206–207

- idauser.cfg* file, 203
- plugins.cfg* file, 323
- connect function, 129
- console configuration file, 206–207
- console version of IDA, 187–195
  - common features, 188–189
  - Linux console specifics, 190–192
  - OS X systems, 192–195
- console-mode debugging, 545–547
- constructor, actions, 161
- CONTEXT structure (Windows), 422*n*, 424
- context-sensitive menus, 62
  - for hex display, 72
  - for IDA summary view, 97–98
- Continue command, in IDA
  - debugger, 504
- control flow
  - hiding, 422
  - and recursive descent assembly, 11
- control flow graphs, 169
- Cooper, Jeremy, 191
- core fonts project, 197
- corrupt database files, 49
- C\_PREDEFINE\_MACROS option, in *ida.cfg* file, 203
- CPU instruction pointer, and instruction classification, 11
- CPU instructions, undocumented, 112–113
- CPU type, PE header indicating target, 341
- Create a new segment dialog, 343
- Create structure/union dialog, 144
- CreateArray function, 256, 257, 295, 557
- CREATE\_BACKUPS option, in *ida.cfg* file, 202
- create\_filename\_cmt function, 353
- CreateNetnode function, 325–326
- CreateThread function, 454
- cross-references, 168–176
  - determining type of, 306
  - display window, 174
  - enumerating, 269–271, 306–308
  - generating from emulator, 377
  - graphs
    - custom, 181–184
    - legacy, 180–181
  - lists, 173–175
    - as navigational targets, 83
    - in text view, 67
- current instruction location, 372

- current position indicator, 55, 57
- custom assembly code, 91
- custom cross-reference graphs, 181–184
- custom\_ana notification code, 393
- custom\_emu notification code, 393
- custom\_mnem notification code, 393
- custom\_out notification code, 393
- custom\_outop notification code, 393
- Cywin tools for Windows, 16, 17

## D

- D, in *nm* utility output, 21
- DarunGrim plug-in, 468
- data
  - carousel, 123, 145
  - converting to code, 121–122
  - cross-references, 168, 171–173, 460
    - IDC functions for, 265
    - SDK functions for, 303–304
  - instructions mixed with, 8, 10
  - manipulation, IDC functions for, 265–266
  - structures
    - layout of, 74
    - recognizing use, 131–142
    - transformations, 122–128
    - specifying data sizes, 123–124
- data displays, 55, 62–71
  - Enums window, 75
  - Exports window, 73
  - Function Calls window, 77–78
  - Functions window, 74
  - Hex View window, 72
    - binary search of content portion, 101
    - synchronized with disassembly view, 72
  - IDA text view, 66–67
  - IDA View-EIP disassembly window, 501–502
  - IDA View-ESP window, 502
  - IDA-View data display, 55, 62–67
    - graph view for, 63–66
    - opening multiple, 66
    - synchronized with hex view, 72
  - Imports window, 73
  - message window, 62, 69
    - in console-mode IDA, 188
    - dumping list of allocated heap blocks to, 454
    - in IDA Desktop, 56

- data displays, *continued*
  - Names window, 55, 62, 68–69
    - adding name to, 106
    - in IDA Desktop, 56, 57
  - Problems window, 78
  - Segments window, 75–76
  - Signature window, 76–77
  - Strings window, 55, 62, 70–71
    - in IDA Desktop, 56, 57
    - refreshing content, 442
  - Structures window, 74–75, 143
  - tool tips, 130*n*
  - Type Libraries window, 77
- data flow analysis, 462–463
- DATA XREF, 169
- database addresses, dumping in file, 453
- database events, 315
- database files, 49–54
  - automated synchronization of
    - changes for multiple users, 485
  - vs. binary files, 238
  - changing word in, 239
  - closing, 52–53
  - corrupt, 49
  - creating, 50–51
  - discarding changes, 53
  - emulator utilization of, 444
  - functions for accessing flags for
    - address in, 385
  - IDA debugger and, 523–524
  - making changes to, 57–58
  - opening, and loading plug-in, 312
  - patching limitations, 240–241
  - reopening, 53–54
  - restoring after crash, 53
  - SDK functions for access, 298–299
  - searching, 100–102
    - IDC functions for, 266–267
    - text searches, 101
  - version incompatibility, 490
- database names manipulation
  - IDC functions for, 262–263
  - SDK functions for, 300
- database segments, emulator creation
  - of, 445
- database-patching menu, 204
- datatypes, 122, 296–298
  - associating with variable, 130
  - information in initial analysis, 51
- db, 100, 123
- dbg.hpp* file, 286, 517
- dbg.py* file (IDA Sync), 486, 487
- dd, 100, 123
- de-obfuscation. *See also* obfuscated code analysis
  - emulation-oriented, 443–455
  - emulator-assisted, 448–453
  - mark for end, 522
  - scripted-oriented, 438–443
- dead listing, 24, 81
- Debug application setup dialog, 548–549
- debug binaries, vs. release binaries, 412–414
- Debug menu
  - ▶ Open Subviews, 545
  - ▶ Show App Screen, 546
- debug registers (x86), 424*n*
- debugger events, 315
- Debugger menu
  - ▶ Attach to Process, 499
  - ▶ Debugger Options, 538
  - ▶ Module List, 503
  - ▶ Process Options, 548
  - ▶ Run, 499
  - ▶ Run to Cursor, 499
  - ▶ Stack Trace, 511
  - ▶ Start Process, 499, 500
  - ▶ Take Memory Snapshot, 524
  - ▶ Tracing, ▶ Instruction Tracing, 508
- Debugger setup dialog, 538–539
- debugger-assisted de-obfuscation,
  - background, 522–523
- debuggers, 15. *See also* IDA debugger
  - console-mode, 545–547
  - detection, 435–436
  - function naming scheme, 531
  - generating listings within, 7
  - hiding, 533–538
  - preventing, 436–437
  - redesign in version 5.3, 574
  - remote, 547–550
  - script to launch, 528–529
- debugging information, *objdump* to
  - display, 24
- DECISION problem, 78
- declarations, 67
  - parsing C structure, 150–151
- decompilers, 5
  - Hex-Rays, 480
- decompression, simple loops, 526–530
- decryption, simple loops, 526–530
- dedicated frame pointer, 94

- deep inspection tools, 27–29
- .def* file extension, 389
- default magic file, 16
- default name, of variable,
  - reverting to, 105
- DEFAULT\_FILE\_FILTER option, 206
- DEFCON Capture the Flag binary, code
  - extraction from, 275
- #define directive, 114, 151, 254
- DelArrayElement function, 257, 557
- DelBpt function, 512, 557
- DelCodeXref function, 557
- DelConstEx function, 557
- del\_dref function, 570
- DelEnum function, 557
- DeleteAll function, 557
- DeleteArray function, 257, 557
- deleting
  - functions, 116
  - netnodes, 295–296
- DelExtLnA function, 557
- DelExtLnB function, 557
- DelFixup function, 557
- DelFunction function, 557
- DelHashElement function, 557
- DelHiddenArea function, 557
- DelHotkey function, 557
- DelLineNumber function, 557
- DelSelector function, 557
- DelSourceFile function, 557
- DelStruc function, 557
- DelStrucMember function, 557
- DelXML function, 557
- Demangle function, 557
- Demangled C++ Names dialog, 162
- demonstration copy of IDA, 33
- demo\_stackframe function, disassembly, 96
- destructor, 160*n*, 161
- desynchronization of disassembly,
  - 418–421, 425
- DetachProcess function, 557
- detail view, for function’s stack frame, 95
- device drivers, platform dependence of
  - code, 7
- Dfirst function, 265, 557
- DfirstB function, 265, 557
- dialogs, 39
  - creating with SDK, 331–335
- DIF files, IDA-generated, 244
- diff utility, 466
- DiffingSuiteSetup.exe*, 467
- directories. *See also individual directory names*
  - for created loader binaries, 355
  - for debugging process setup, 549
  - IDA layout, 37–39
  - for plug-ins, 319–320
  - for SDK, 281–283
- disassemblers, 5, 28–29
- disassembly
  - basics, 5
  - cross-reference information in
    - listing, 169
  - desynchronization, 418–421
    - warning during, 529
  - determining where to begin, 9
  - line components, 111–113
    - IDC functions for, 267
  - options controlling lines, 202
  - output lines in listing, 380
  - process, 7–14
    - basic algorithm, 8–9
    - linear sweep disassembly, 9–11
    - recursive descent assembly, 11–14
  - purpose of tools, 6–7
  - structure notation for readability, 150
  - techniques to prevent, 522
  - theory, 4
  - window, 55. *See also* IDA-View data display
    - in console IDA, 188
- disassembly manipulation
  - basic code transformations, 110–122
    - code display options, 111–113
    - converting data to code, 121–122
    - formatting instruction operands, 114–115
    - manipulating functions, 115–121
  - basic data transformations, 122–128
    - arrays, 126–128
    - specifying data sizes, 123–124
    - strings, 124–126
  - comments, 108–110
  - names, 104–108
- discarding changes to database files, 53
- disclosure, of vulnerability to public, 465
- discovery, of vulnerability, 465
- diskio.hpp* file, 351, 362
- dispatcher function, 299
- DISPLAY\_COMMAND\_LINE option, 204, 206, 251
- DISPLAY\_PATCH\_SUBMENU option, 204, 206
- displays. *See* data displays
- diStorm utility, 28

*dll2idt.exe*, 232  
 Dnext function, 265, 557  
 DnextB function, 265, 557  
 documentation  
   for IDA Pro, 34  
   searching through, 284  
 dollar sign (\$), in AT&T assembly  
   syntax, 9  
 dont\_use\_sprintf, 285  
*dos.ldw*, 46  
 double word, 100  
 double-click navigation, 82–83  
 d\_out function, 384  
 downloading, IDA purchased copy, 34  
 DR0–7 registers, 505  
 dummy names, 104. *See also* auto-generated names  
 dumpbin utility, 23, 25  
   for obfuscation, 428  
 “dup” construct, 128  
 dw, 100, 123  
 Dword function, 259, 439, 557  
 dword\_xxxxxx autogenerated names, 69  
 dynamic  
   analysis, 6, 433  
   linking, 22–23  
   memory allocation, for heap-allocated arrays, 135  
 dynamically computed target addresses, 421–427  
 dynamically linked binary, 178  
 dynamic\_cast operator, 164

## E

EAX register, 11–12, 93  
 EBP (extended base pointer)  
   register, 93  
 ECX register, 93  
   use in compiled C++, 157  
 Edit Function dialog, 118  
 Edit menu  
   ▶ Comments, 108  
   ▶ Expand Struct Type, 146  
   ▶ Functions  
     ▶ Create Function, 116  
     ▶ Delete Function, 116  
   ▶ Patch Program, 204, 238–241  
     ▶ Assembler, 239  
     ▶ Change Byte, 238–239  
   ▶ Plugins, 316

▶ Segments  
   ▶ Create Segment, 343, 345  
   ▶ Move Current Segment, 344  
   ▶ Rebase Program, 341  
   ▶ Shrink Struct Type, 146  
   ▶ Strings, 124  
   ▶ Undefine, 121–122  
 EDX register, 93  
 eEye Digital Security, 467–468  
 elements, number in array, 127  
 ELF. *See* Executable and Linking Format (ELF)  
 #else directive, 255  
 embedded strings, searching for, 27, 101  
 emu function, 376  
*emu.cpp* file, 376  
 Emulate menu ▶ Windows ▶ Set Import Address Save Point, 453  
 emulation-oriented de-obfuscation, 443–455  
 emulator  
   database segments creation, 445  
   database utilization, 444  
   output generation in message window, 452  
   in processor module, 366, 376–379  
   for Python processor, 379  
 EnableBpt function, 557  
 EnableTracing function, 514, 558  
 encryption, of obfuscated programs, 525  
 end address, of functions, 118  
 endEA data member, 296  
 #endif directive, 255  
 entry points. *See* main function; program entry point  
*entry.hpp* file, 286, 362  
 enumerated constants set, defined, 368  
 Enums window, 75  
 epilogue, of function, 87  
 Erdelyi, Gergely, 481  
 error handling, in IDC scripts, 255–256  
 ESC key, 62, 85  
*etc* directory, for SDK, 282  
 Eval function, 558  
 event notification, in plug-ins, 315–316  
 exception handlers  
   in remote debugging, 550  
   tracing, 540–542  
   in Windows-oriented malware, 422–424  
 Exception handling dialog, 539

- exceptions
  - in IDA debugger, 538–544
  - intentionally generating, 436
  - passing to application, 540
- Exceptions configuration dialog, 539
- exceptions.cfg* file, 539
- exclusions files, 222
- EXE files, IDA-generated, 243–244
- Exec function, 558
- Executable and Linking Format (ELF), 8
  - file command information on, 17
  - loader, 395
  - search for instructions, 472
- executable files, obfuscation, 19
- execute breakpoints, 506
- execution traces, 508
- exe.sig* file, 405
- Exit function, 558
- exploit-development process, 469–475
  - finding useful virtual addresses, 473–475
  - instruction sequences location, 472–473
  - for patched vulnerability, 466
  - stack frame breakdown, 470–472
- export ordinal number, 73*n*
- exported functions, enumerating, 272
- Exports window, 73
- expressions
  - for IDA breakpoint conditions, 507–508
  - in IDC scripts, 252
- expr.hpp* file, 286, 325, 575
- extended base pointer (EBP) register, 93
- extlang\_t structure, 575
- ExtLinA function, 558
- ExtLinB function, 558

**F**

- Falliere, Nicolas, 436, 533–534, 536, 537
- far address for flow, 169
- far functions, 119
- Fast Library Identification and Recognition Technology. *See* FLIRT
- fastcall convention (x86), 89–90
- Fatal function, 558
- fclose function, 261, 570
- fgetc function, 262, 570
- field names, in structures, 136
- fields, adding to structure, 145
- file analysis, for unknown file format, 338–339
- file command, 16–18
  - information on ELF, 17
  - limitations, 18
- file extensions, configuration, 203
- file headers, loading, 153
- file input/output, IDC functions for, 261–262
- File menu
  - ▶ IDC Command, 250
  - ▶ IDC File, 250
  - ▶ Load File
    - ▶ FLIRT Signature File, 214, 223
    - ▶ IDS File, 233
    - ▶ Parse C Header File, 151
  - ▶ Produce File, 241
    - ▶ Create C File, 480
    - ▶ Create EXE File, 347
- file pointers, and IDA, 354
- FILE stream, 354
- file utility, 218–219
- file2base function, 352, 353, 358
- FileAlignment field, 342
- FILE\_EXTENSIONS option, 205, 206
- filelength function, 262, 570
- file-loading dialog, 337
- Filemon utility, 435
- filename extensions, 16
- files, listing entry points into, 73
- filetypes
  - associating extensions with, 205
  - identifying, 16, 45
- FindBinary function, 266, 558
- FindCode function, 266, 269, 558
- FindData function, 266, 558
- FindExplored function, 558
- FindFuncEnd function, 558
- FindImmediate function, 558
- FindSelector function, 558
- findStackBuffers function, 463–465
- FindText function, 266, 558
- FindUnexplored function, 558
- FindVoid function, 558
- first-generation programming languages, 4
- FirstFuncFchunk function, 558
- FirstSeg function, 558
- fixed-argument functions, stdcall convention for, 89
- fix\_proc utility, 390
- flagCalls function, 462

- flags, in Functions window, 74
- flags field
  - for loader module, 349
  - for plug-ins, 311
- FLAIR. *See* FLIRT
- flair52.zip* file, 216
- Flake, Halvar, 186, 463
- FLIRT (Fast Library Identification and Recognition Technology), 38, 212
  - applying signatures, 212–216
  - collisions
    - resolution for, 223
    - in signature generation, 221
    - creating signature files, 216–225
  - `f_loader` file type, 395
  - floating-point values, in IDC scripts, 251
  - flowcharts, legacy, 177–178
  - flows, 169–171
    - colored arrows for, 64
  - Follow TCP Stream command (Wireshark), 476
  - FontForge, 197
  - fonts, 58
    - for Wine, 197
  - `fopen` function, 261, 570
  - for loops, 252
  - forking new project, in CollabREate, 491
  - `form` function, 261, 570
  - formatting, removing, 121
  - forms, creating customized with SDK, 331–335
  - forums, 35
  - 4-byte overwrite, 474
  - fourth-generation programming languages, 4
  - `fprintf` function, 262, 570
  - fpro.h* file, 286
  - `fputc` function, 262, 570
  - frame pointer, 86*n*
    - dedicated, 94
    - delta, 119
  - frame.hpp* file, 286, 300
  - FreeType, 197
  - freeware version of IDA, 32–33, 551–553
    - restrictions, 552
  - `fseek` function, 570
  - `ftell` function, 570
  - funcs.hpp* file, 286, 300
  - `func_t` class, 296
  - Function Calls window, 77–78
  - function-oriented control flow graph, 184
  - functions, 85, 115–121
    - addresses, array as import table, 452
    - argument identification in initial analysis, 51
    - attributes of, editing, 117–120
    - augmenting information, 228–234
    - call graphs, 77, 169, 178–180
    - call instructions, 12
    - call tree, 77
    - calls, 86
      - cross-reference listing for, 175
    - chunks, 116–117, 296
    - comments, 110
    - creating, 115–116
    - debugger naming scheme for, 531
    - deleting, 116
    - enumerating, 268, 304–305
    - epilogue of, 87
    - finding callable address, 460–461
    - IDC mapped to SDK, 555–572
    - information in *.ids* files, 230
    - locating all with stack-allocated buffers, 463–465
    - in Name window, 68
    - neighbors of, 77
    - new in version 5.3, 574
    - overloading, mechanism for differentiating versions, 25
    - overloaded versions of, 162
    - parameter placement on stack, 87
    - prologue of, 87
    - renaming based on signature, 214–215
  - return instructions, 13–14
  - in SDK, 298–304
    - code cross-references, 303
    - data cross-references, 303–304
    - for database access, 298–299
    - database names manipulation, 300
    - function manipulation, 300–301
    - for manipulating, 300–301
    - segment manipulation, 302–303
    - structure manipulation, 301
    - user interface, 299–300
  - storing addresses of dynamically linked library, 474
  - tails, 117
  - tracing, 508
  - type signatures, generating, 229

Functions window, 74  
fuzzing, 6*n*

## G

Gaobot worm, 19  
garbage collection, 52  
gcc/Cygwin binary, startup routine  
    from, 408  
gdb (GNU debugger), 11, 540  
    ptrace API use, 437  
GDL (Graph Description  
    Language), 176  
*gdl.hpp* file, 286  
GenCallGdl function, 558  
General Registers window, 502–503  
GenerateFile function, 558  
GenFuncGdl function, 558  
GetArrayElement function, 257, 558  
GetArrayId function, 256, 558  
GetBmaskCmt function, 558  
GetBmaskName function, 558  
GetBptAttr function, 513, 559  
GetBptEA function, 512, 559  
GetBptQty function, 512, 559  
get\_byte function, 298  
GetCharPrm function, 559  
GetColor function, 559  
GetCommandLine function, 410, 411  
GetConstBmask function, 559  
GetConstByName function, 559  
GetConstCmt function, 559  
GetConstEnum function, 559  
GetConstEx function, 559  
GetConstName function, 559  
GetConstValue function, 559  
GetCurrentLine function, 559  
GetCurrentThreadId function, 559  
GetDebuggerEvent function, 514–515, 559  
GetDisasm function, 267, 559  
GetEntryOrdinal function, 272, 559  
GetEntryPoint function, 272, 559  
GetEntryPointQty function, 272, 559  
GetEnum function, 559  
GetEnumCmt function, 559  
GetEnumFlag function, 559  
GetEnumIdx function, 559  
GetEnumName function, 559  
GetEnumQty function, 559  
GetEnumSize function, 559  
GetEnvironmentStrings Windows API  
    function, 411  
GetEvent function, 514  
GetEventBptHardwareEa function, 559  
GetEventEa function, 560  
GetEventExceptionCode function, 560  
GetEventExceptionEa function, 560  
GetEventExceptionInfo function, 560  
GetEventExitCode function, 560  
GetEventId function, 560  
GetEventInfo function, 560  
GetEventModuleBase function, 560  
GetEventModuleName function, 560  
GetEventModuleSize function, 560  
GetEventPid function, 560  
GetEventTid function, 560  
GetFchunkAttr function, 560  
GetFirstBmask function, 560  
GetFirstConst function, 560  
get\_first\_cref\_from function, 303  
get\_first\_cref\_to function, 303  
get\_first\_dref\_from function, 303  
get\_first\_dref\_to function, 303  
GetFirstHashKey function, 560  
GetFirstIndex function, 560  
GetFirstMember function, 560  
GetFirstModule function, 560  
GetFirstStrucIdx function, 560  
GetFixupTgtDispl function, 560  
GetFixupTgtOff function, 560  
GetFixupTgtSel function, 560  
GetFixupTgtType function, 560  
GetFlags function, 560  
GetFpNum function, 560  
GetFrame function, 268, 560  
get\_frame function, 301  
GetFrameArgsSize function, 560  
GetFrameLvarSize function, 471, 560  
GetFrameRegsSize function, 471, 560  
GetFrameSize function, 561  
get\_func function, 300  
getFuncAddr function, 460–461  
get\_func\_name function, 300  
GetFuncOffset function, 561  
get\_func\_qty function, 300  
GetFunctionAttr function, 263, 269, 561  
GetFunctionCmt function, 561  
GetFunctionFlags function, 274, 561  
GetFunctionName function, 263, 561  
GetHashLong function, 561  
GetHashString function, 561  
GetIdaDirectory function, 561  
GetIdbPath function, 561  
GetInputFile function, 272, 561

GetInputFilePath function, 561  
 GetLastBmask function, 561  
 GetLastConst function, 561  
 GetLastHashKey function, 561  
 GetLastIndex function, 561  
 GetLastMember function, 561  
 GetLastStrucIdx function, 561  
 GetLineNumber function, 561  
 GetLocalType function, 561  
 GetLocalTypeName function, 561  
 get\_long function, 298  
 GetLongPrm function, 561  
 getmainargs library function, 409  
 GetManualInsn function, 561  
 get\_many\_bytes function, 298  
 GetMarkComment function, 561  
 GetMaxLocalType function, 561  
 get\_member function, 301  
 GetMemberComment function, 562  
 GetMemberFlag function, 562  
 GetMemberName function, 562  
 GetMemberOffset function, 268, 562  
 GetMemberQty function, 562  
 GetMemberSize function, 464, 562  
 GetMemberStrId function, 562  
 GetMnem function, 267, 562  
 GetModuleName function, 562  
 GetModuleSize function, 562  
 get\_name function, 300  
 get\_name\_ea function, 300  
 getn\_area function, 304  
 GetnEnum function, 564  
 get\_next\_area function, 304  
 GetNextBmask function, 562  
 GetNextConst function, 562  
 get\_next\_cref\_from function, 303  
 get\_next\_cref\_to function, 303  
 get\_next\_dref\_from function, 303  
 get\_next\_dref\_to function, 303  
 GetNextFixupEA function, 562  
 GetNextHashKey function, 562  
 GetNextIndex function, 562  
 GetNextModule function, 562  
 GetNextStrucIdx function, 562  
 getn\_next\_func function, 300  
 getnseg function, 302  
 getopcode.c program, 472  
 GetOperandValue function, 267, 562  
 GetOpnd function, 267, 562  
 GetOpType function, 267, 562  
 get\_original\_byte function, 298  
 GetOriginalByte function, 563  
 get\_original\_long function, 299  
 get\_original\_word function, 299  
 GetPrevBmask function, 563  
 GetPrevConst function, 563  
 GetPrevFixupEA function, 563  
 GetPrevHashKey function, 563  
 GetPrevIndex function, 563  
 GetPrevStrucIdx function, 563  
 GetProcAddress function, 428, 430, 451, 452*n*, 530  
 GetProcessName function, 563  
 GetProcessPid function, 563  
 GetProcessQty function, 563  
 GetProcessState function, 563  
 GetReg function, 563  
 get\_reg\_val function, 519  
 GetRegValue function, 507, 512, 563  
 get\_screen\_ea function, 300  
 getseg function, 302  
 GetSegmentAttr function, 563  
 get\_seg\_by\_name function, 302  
 get\_seg\_name function, 302, 303  
 get\_seg\_qty function, 302  
 GetShortPrm function, 563  
 GetSourceFile function, 563  
 GetSpd function, 563  
 GetSpDiff function, 563  
 GetString function, 563  
 GetStringType function, 563  
 get\_struct function, 301  
 GetStructComment function, 563  
 get\_struct\_id function, 301  
 GetStructId function, 563  
 GetStructIdByName function, 563  
 GetStructIdx function, 563  
 GetStructName function, 563  
 GetStructNextOff function, 563  
 GetStructPrevOff function, 563  
 GetStructQty function, 563  
 GetStructSize function, 268  
 get\_struct\_size function, 301  
 GetStructSize function, 563  
 GetThreadId function, 563  
 GetThreadQty function, 563  
 GetTrueName function, 563  
 GetTrueNameEx function, 564  
 get\_true\_seg\_name function, 303  
 GetType function, 564  
 GetVxdFuncName function, 564  
 get\_word function, 298  
 GetXML function, 564  
 Gigapede, 523

- gl\_comm global variable, 382–383
- global arrays
  - and netnodes, 295
  - in offset cross-references, 173
  - possible uses for, 257
- global offset table (GOT), 474
- global persistent arrays, 256
- global variables, 69
  - formatting as structures, 150
  - IDC and, 252
- globally allocated arrays, 132–133
- globally allocated structures, 138
- Gnome gnome-terminal, 191
- gnome terminal, 193
- GNU
  - Assembler, 9
  - binutils tool suite, 24
  - debugger (gdb), 11, 540
    - ptrace API use, 437
  - gcc/g++ compiler, 88–89, 90
    - vs. Microsoft Visual C/C++ compiler, 161
    - and pack pragma, 137
    - regparm keyword in, 91
- Go button, in welcome screen, 45
- GOT (global offset table), 474
- Graph Description Language (GDL), 176
- graph modes
  - user interface, 184–185
  - viewing cross-reference comments in, 185
- Graph Overview window, 64
- graph theory, 168
- graph view
  - for disassembly view, 55, 63–66
  - line prefixes in, 65
  - rearranging blocks within, 66
  - switching between text view and, 184
- graph view node, 185
- graphical user interface. *See* GUI
- graphing, 176–186
  - custom cross-reference graphs, 181–184
  - integrated graph view, 184–186
  - legacy call, 178–180
  - legacy cross-reference, 180–181
  - user xref charts, 181–183
- GRAPH\_VISUALIZER option, in *ida.cfg* file, 202
- grep, 284
- grouping blocks, 66
- grouping nodes within graph, 185–186
- Gudmundsson, Atli Mar, 244
- GuessType function, 564
- GUI (graphical user interface), 39
  - default behavior, 203
  - on non-Windows platforms, 196–197
- Guilfanov, Ilfak, 31, 530, 540, 542
  - blog of, 35
- gunzip, 37

## H

- Hall of Shame, 32
- hardware abstraction layers,
  - detection, 434
- hardware-assisted breakpoints, 505–507, 526
- hash value, resolving function names with, 431
- hashset function, 294
- hashstr function, 294
- hashval function, 294
- hashval\_long function, 294
- hashvals, 291, 294
- hasName function, 570
- hasValue function, 570
- HBGary, 458
- head command, 212*n*
- header files, 280, 284–288
  - in C, parsing, 151–152
- headers, objdump to display, 23
- .headers program segment, 445
- heap blocks, dumping list to message window, 454
- .heap database segment, 445
- heap-allocated
  - arrays, 135–136
  - objects, destructors, 161
  - structures, 139–141
- help, for IDA Pro, 34
- helper function, to format and output instruction operand, 380
- HELPPFILE option, 203–204
- hex dumps, 189
- Hex-Rays, 31, 573
  - demo version of IDA, 551
  - download site, 227, 479
  - free version of IDA, 551–553
  - restrictions, 552
  - Research & Resources forum, 283
  - submitting bug reports, 58
- Hex-Rays decompiler, 480–481

- Hex View window, 72
  - binary search of content portion, 101
  - synchronized with disassembly view, 72
- hexadecimal constants, 114
- hexadecimal values, and navigation, 83
- HideArea function, 564
- HideDebugger.idc* script, 538
- hiding debuggers, 533–538
- HighVoids function, 564
- HKEY\_CURRENT\_USER\Software\Hex-Rays\IDA, 44, 207
- hook\_to\_notification\_point function, 315
- hostname, for remote debugging, 549
- hotkeys, 39
  - associating scripts with, 258
  - in console IDA, 188
  - for Linux, 190
  - for IDA Sync, 486
  - mapping, 203
- .hpp* file extension, 284
- HTML files, IDA-generated, 245

## I

- .id0* file, 49
- .id1* file, 49
- IDA Application Programming Interface, 284–308
  - header files, 284–288
  - iteration techniques, 304–308
    - enumerating cross-references, 306–308
    - enumerating functions, 304–305
    - enumerating structure members, 305–306
  - netnodes, 288–296
    - creating, 289–291
    - data storage, 291–295
    - declaring, 290
    - deleting, 295–296
- IDA Colors dialog, 208
- IDA debugger, 497–519
  - automating tasks, 512–519
    - with plug-ins, 517–519
    - scripts with IDC, 512–517
  - databases and, 523–524
  - displays, 501–503
  - exceptions, 538–544
  - instruction pointer warning, 529
  - launching, 498–500
    - for obfuscated code, 525–544
  - process control, 504–511
    - breakpoints, 505–508
    - stack traces, 511
    - tracing, 508–510
    - watches, 511
  - shortcoming, 540
- IDA Desktop, 54–56
  - during initial analysis, 56–58
  - tips and tricks, 58
- IDA freeware, 32–33, 551–553
  - restrictions, 552
- IDA loader modules, 51
- IDA Options dialog
  - Analysis tab, 388
  - Disassembly tab, 111
  - Graph tab, 56
  - Strings tab, 125
- IDA Palace, 35
- IDA Plug-in Writing in C/C++ (Micallef), 283
- IDA Pro (Interactive Disassembler Professional)
  - basics, 31–32
  - crashes, database restore after, 53
  - directory layout, 37–39
  - file loading, 46–47
    - and file pointers, 354
  - installation, 35–39
  - launching, 44–46
  - loader modules, 347
  - new features in version 5.3, 573–575
  - obtaining, 32–34
  - purchasing, 33–34
  - support resources, 34–35
  - upgrading, 34
  - versions, 33
- IDA Software Development Kit (SDK)
  - basics, 279–283
  - collabREate plug-in integration, 490
  - configuring build environment, 283
  - creating customized forms, 331–335
  - datatypes, 296–298
  - directory structure, 281–283
  - functions, 298–304
    - code cross-references, 303
    - data cross-references, 303–304
    - for database access, 298–299
    - database names manipulation, 300
    - function manipulation, 300–301
    - segment manipulation, 302–303

- structure manipulation, 301
  - user interface, 299–300
- installing, 281
- update in version 5.3, 574
- IDA stack views, 95–100
- IDA Sync, 485–488
- IDA Text view, 66–67
- IDA View-EIP disassembly window, 501–502
- IDA View-ESP window, 502
- idaadv* directory, 37
- ida.cfd* file, 207
- ida.cfg* file, 202–203
- idag.exe*, 36, 189
  - running in batch mode, 195
- idag64.exe*, 189
  - running in batch mode, 195
- idagui.cfg* file, 202, 203–206, 238, 251
  - hotkey configuration, 204–205
- ida.hlp*, and *loadint.exe*, 235
- ida.hpp* file, 284, 286, 297
- ida.idc* file, 258
- idaidp.hpp* file, 366
- idainfo struct, 297
- ida.int* file, 234
- ida.key* file, 32, 36, 190
  - copying to Mac, 192
- idal* executable file, 37
  - setting permissions and ownership, 546–547
- idaldx.h* header file, 351
- idamake.pl* script, 283, 318
- IDAPython, 481–484
- IDARub, 484–485
- idarub.cpp* file, 484
- idastd* directory, 37
- ida\_sync\_server.py* file, 486
- idatui.cfg* file, 202, 206–207
- idauser.cfg* file, 203
- IDA-View data display, 55, 62–67
  - graph view for, 63–66
  - opening multiple, 66
  - synchronized with hex view, 72
- idaw.exe*, 36, 189
  - running in batch mode, 195
- idaw64.exe*, 189
  - running in batch mode, 195
- ida-x86emu (x86emu) plug-in, 336, 444–445, 492
  - additional features, 453–454
  - and anti-debugging, 454–455
  - breakpoints, 446
  - control dialog, 445
  - emulator-assisted de-obfuscation, 448–453
  - functions emulated by, 451
  - initializing, 445–446
  - operation, 446–448
- .idb* files, 49, 52, 156
- IDB\_2\_PAT utility, 221
- IDC command dialog, 254
- idc* directory, 38
- IDC language, 251–257
  - command-line option, 251
  - error handling, 255–256
  - expressions, 252
  - functions, 253–254, 258–267
    - for code cross-references, 264
    - for data cross-references, 265
    - data manipulation, 265–266
    - database names manipulation, 262–263
    - database search, 266–267
    - disassembly line components, 267
    - file input/output, 261–262
    - functions dealing with, 263
    - mapped to SDK, 555–572
    - for reading and modifying data, 259–260
    - string manipulation, 261
    - user interaction, 260
  - persistent data storage, 256–257
  - plug-ins for extending, 324–327
  - programs, 254–255
  - scripting debugger actions, 512–517
  - statements, 252–253
  - variables, 251–252
- idc.idc* file, preprocessor directive to include, 254
- idc\_value\_t* class, 297
- idc\_value\_t* SDK datatype, 326
- Identifier search, 101
- idp.hpp* file, 286, 367, 385, 393
- IDP\_INTERFACE\_VERSION constant, 310
- ids* directory, 38
- .ids* files, 228
  - basics, 230–231
  - creating, 232–234
- IDS utilities, 228
- idsnames* text file, 233
- idsutils utilities, 230, 232

- .idt* files, 230
  - script to generate, 272
  - syntax for, 232
  - zipids.exe* utility to compress, 233
- `#ifdef` directive, 255
- Ignore option, for user xrefs chart, 183
- Ifak. *See* Guilfanov, Ifak
- .ilx* file extension, 389
- IMAGE\_DOS\_HEADER, 153–155, 340
- IMAGE\_NT\_HEADERS structure, 153, 340, 342
- IMAGE\_SECTION\_HEADER template, 342
- import address save point, 453
- import tables, 73*n*
  - adjusting names, 532
  - function addresses in array as, 452
  - reconstructing, 431, 530–533
  - validation on, 50
- imported function table, restoration, 523
- imported functions
  - editing, 121, 231
  - obfuscation, 428–432
- imported name, 68
- importing structures, 150–152
- `import_node` netnode, 288
- Imports window, 73
- `import_type` function, 357
- ImpREC (Import REConstruction)
  - utility, 523
- INC (include) files, 243
- `#include` directive, 254
  - and parsing header files, 151
  - resolving dependencies, 203
- include* directory, for SDK, 282
- include header files, 284
- Indent function, 564
- INDENTATION option, in *ida.cfg* file, 202
- indentation, with `MakeLine` function, 382
- index value, for individual array
  - elements, 131
- indexes, for array, 128
- indirect code paths, recursive
  - descent and, 13
- `inf` variable, 297
- inheritance relationships, 164–165
- init function pointer, in `plug-in_t`
  - class, 311
- The initial autoanalysis has been finished*
  - message, 57
- initializing
  - objects, 161
  - plug-ins, 313–315
- `init_loader_options` function, 350, 352
- inline functions, 165
- input file, for debugging process
  - setup, 549
- ins.cpp* file, 367
- ins.hpp* file, 368
- `insn_t` class, 297–298, 371
  - processor-dependent fields in, 374
- installing
  - eEye Digital Security, 467
  - FLAIR utilities, 216
  - IDA Pro, 35–39
    - on OS X and Linux, 37
    - on Windows, 36
  - IDA Software Development Kit, 281
  - plug-ins, 322–323
  - third-party plug-ins, 480
- install\_make.txt* file, 283
- install\_visual.txt* file, 320
- `instruc_t` structures, array of, 367
- instructions
  - emulator. *See* emulator
  - enumerating, 268
  - formatting operands, 114–115
  - locating sequences, 472–473
  - mixing with data, 8, 10
  - operands, 373
  - tracing, 508
- instrumentation, detection, 435
- `int 0x80` instruction, 91
- `int 3` instruction, 505
- integers
  - formatting constants, 114
  - in IDC scripts, 251
- integrated graph view, 184–186
- Intel 64 and IA-32 Architectures Software Developer’s Manual, 507
- Intel format, 8–9
- Interactive Disassembler Professional.
  - See* IDA Pro (Interactive Disassembler Professional)
- interrupt-handling routine, return
  - from, 419*n*
- `invoke_callbacks` function, 386
- `iret` instruction (x86), 419*n*
- `isBin0` function, 570
- `isBin1` function, 570
- `IsBitfield` function, 564
- `isChar0` function, 570
- `isChar1` function, 570
- `isCode` function, 570
- `isData` function, 570

- IsDebugged field, of process environment block, 534
- IsDebuggerPresent function, 436, 451, 534
- isDec0 function, 571
- isDec1 function, 571
- isDefArg0 function, 571
- isDefArg1 function, 571
- isEnum0 function, 571
- isEnum1 function, 571
- IsEventHandled function, 564
- isExtra function, 571
- isFlow function, 571
- isFop0 function, 571
- isFop1 function, 571
- isHead function, 571
- isHex0 function, 571
- isHex1 function, 571
- isLoaded function, 259, 260, 299, 571
- isOct0 function, 571
- isOct1 function, 571
- isOff function, 377
- isOff0 function, 571
- isOff1 function, 571
- isRef function, 571
- isSeg0 function, 571
- isSeg1 function, 571
- isStkvar0 function, 571
- isStkvar1 function, 571
- isStroff0 function, 571
- isStroff1 function, 571
- isTail function, 571
- IsUnion function, 564
- isUnknown function, 571
- isVar function, 571
- Itanium processor, IDA support for, 33
- ItemEnd function, 564
- ItemSize function, 564
- iTERM, 193

## J

- ja (jump above) instruction, 402
- JAL instruction (MIPS), 86
- Java *.class* file, magic numbers to identify, 16
- Java loader, 361
- jmp eax instruction, 11
- jmp functions, 412–413
- JPEG image file, magic numbers to identify, 16
- JR instruction (MIPS), 86
- jump flow, 64*n*, 169, 171

- processor flag for changing conditional to absolute, 420
- jump (jmp) functions, 412–413
- Jump function, 260, 564
- Jump menu
  - ▶ Jump to Next Position, 85
  - ▶ Jump to Previous Position, 84
  - ▶ Jump to Problem, 204
- jump tables, 10
  - for different compilers, 400–404
- Jump to address dialog, 84
- Jump to cross-reference dialog, 174
- Jump to Cursor command, in
  - ida-x86emu plug-in, 447
- jumpto function, 300
- junk strings, 71

## K

- KDE konsole, 191
- kernel32.dll* file, 430
- kernwin.hpp* file, 284, 286, 327, 332, 335
- key files, 32
  - for Linux and Mac IDA distributions, 190
  - for upgrade, 34
- keyboard
  - hotkey mappings, 203
  - macro definition syntax, 206
  - preventing system from overriding mappings, on OS X, 194–195
  - zoom control, 64
- known extensions filter, 45

## L

- label, for form input field, 332
- last-known good state of database, 53
- launching
  - IDA debugger, 498–500
  - IDA Pro, 44–46
- layout
  - of binary files, 48
  - of directories, 37–39
  - of local variables, 91
- ldd (list dynamic dependencies) utility, 22–23
  - for obfuscation, 428
- ldr* directory, 282, 350
- LDRF\_RELOAD flag, for loader module, 349
- LDSC loader\_t object, 348
- LDSC object, declaring, 351

- .ldw* file extension, 355
- legacy
  - call graphs, 178–180
  - cross-reference graphs, 180–181
  - flowcharts, 177–178
  - IDA graphing, 176–184
- letter coding, for names, 68
- libbfd (Binary File Descriptor library), 24
- libc.a*, 213
- libc\_FreeBSD61.exc* file, 222
- `_libc_start_main` function, 407
- libraries
  - binary variations of, 213
  - debugging, 550
  - identifying and acquiring static, 217–219
- library functions, 68, 129, 211
  - calling conventions for, 91
  - categorizing, 76
  - flagging function as, 119
  - resolving location, 22
  - Trace Over, 510
- library handles, 451*n*
- libstdc++.so.5 shared library, 37, 190
  - Linux IDA installation and, 547*n*
- libXXX.YYY* directories, for SDK, 282
- license for IDA, 32, 33
- line prefixes
  - customizing, 112
  - in graph view, 65
- LineA function, 564
- linear sweep disassembly, 9–11
- LineB function, 564
- lines.hpp* file, 286, 380, 382–383
- Linux
  - ELF, search for instructions, 472
  - IDA debugger, 546
  - IDA version, installation, 36, 37
  - .ilx* file extension for, 389
  - overwriting executable files, 323
  - plug-in versions for, 483
  - Shiva ELF obfuscation tool for, 437
  - linux\_server*, 547
  - Linux-style command shell, for Windows operating system, 17
- list dynamic dependencies (ldd) utility, 22–23
- List of available library modules
  - dialog, 214
- `list_callers` function, 307–308
- listing view, for disassembly view, 55
- Liston, Tom, 435
- Litchfield, David, 472
- little-endian CPU, 10*n*
- .llx* file extension, 355
- lnames array, 387
- load string byte (lodsbyte) instruction, 441
- LoadDebugger function, 564
- loader modules, 347
  - alternative strategies, 361
  - building, 355
  - creating, 280
  - processor coupled with, 395
  - simpleton file format, 350–355
  - user input to file analysis, 50
  - writing, 348–360
- loader segments, 524
- loader.hpp* file, 287, 310, 313, 362
  - `loader_t` struct layout defined in, 348
- loaders* directory, 38, 46, 347
- `loader_t` object, 348
- `load_file` function, 349, 352–353, 395–396
- loadfile function, 262, 571
- loading
  - plug-ins, 312
  - TIL files, 155
- Loading Offset field, for Binary File input format, 47
- Loading Segment field, for Binary File input format, 47
- loadint utilities, 234–236
- loadint.exe*, and *ida.hpp*, 235
- LoadLibrary function, 428, 430, 530
- LoadLibraryA function, emulated
  - version, 451
- load\_pcap\_file function, 358–360
- LoadTil function, 564
- local name, of named location, 106
- Local Types subview, 150, 151
- local variables, 97
  - identification in initial analysis, 51
  - layout, 91
  - space allocation for, 86
- Local variables area, of functions, 118
- LocByName function, 263, 271, 564
- LocByNameEx function, 263, 564
- loc\_XXXXX autogenerated names, 69
- lodsbyte (load string byte) instruction, 441
- logging, of trace events, 508
- logical addresses, in *.map* files, 242
- loops, 67
- LowVoids function, 564
- LPH object, function pointers, 386–387

LPH structure, initializing, 367–371  
lread function, 352  
lread4bytes function, 351–352  
LST files, 243  
ltoa function, 261, 571

## M

Mach-O loader, 395  
machine languages, 4  
    display options, 113  
Macintosh. *See* OS X systems  
MackT, 523  
MACRO keyword, 207  
    *mac\_server* file, 546–547  
magic numbers, 16  
main function, 254  
    location for different compilers,  
        405–412  
    in C, vs. `_start`, 213  
MakeAlign function, 564  
MakeArray function, 564  
MakeByte function, 266, 564  
MakeCode function, 265, 564  
MakeComm function, 266, 564  
MakeData function, 564  
MakeDouble function, 564  
MakeDword function, 564  
makefiles, 318–319  
    for Python processor, 391–392  
MakeFloat function, 565  
MakeFrame function, 565  
MakeFunction function, 266, 565  
MakeLine function, 381, 382  
MakeLocal function, 565  
MakeNameEx function, 263, 565  
MakeOword function, 565  
MakePackReal function, 565  
MakeQword function, 565  
MakeRptCmt function, 565  
MakeStr function, 266, 565  
MakeStructEx function, 565  
MakeTbyte function, 565  
MakeUnkn function, 265, 565  
MakeUnknown function, 565  
MakeVar function, 565  
MakeWord function, 565  
malloc, 159  
malware, 50  
    analysis, 6  
        goal, 521  
        analysts, 399

    danger of running, 522–523  
    embedded executables, 454  
    emulation vs. debugging, 444  
    exception handler in, 422–424  
    execution environment for, 433  
    infection of debugging machine, 500  
    obfuscation, 19  
    rules for working with, 525–526  
    Windows operating system and, 450  
many-to-many operation,  
    compilation as, 5  
*.map* files, 242  
MarkPosition function, 565  
MASM (Microsoft Assembler), 9  
MaxEA function, 565  
maximum possible size, of array, 127  
MAX\_NAMES\_LENGTH option, in *ida.cfg*  
    file, 202  
MDI (Windows Multiple Document  
    Interface), 331*n*  
mem2base function, 359  
member functions, calls to, 157  
member\_t class, 297, 301  
memory. *See also* stacks, frames  
    layout of binary files, 48  
    references, reformatting for  
        reliability, 147  
    reserving block in emulation  
        heap, 454  
    snapshot, of running process by  
        debugger, 499  
memory modification dialog  
    (x86emu), 448  
Memory organization dialog, 48  
Message function, 260, 565  
message window, 62, 69  
    in console IDA, 188  
    dumping list to of allocated heap  
        blocks to, 454  
    in IDA Desktop, 56  
Metakit embedded database library, 486  
Metasploit project, 472, 475  
methods, 85. *See also* functions  
Micallef, Steve, 284  
    IDA Plug-in Writing in C/C++, 283  
Microsoft, Patch Tuesday cycle of  
    updates, 458  
Microsoft Assembler (MASM), 9  
Microsoft Interface Definition  
    Language (MIDL), 492  
Microsoft Knowledge Base, on  
    WinHelp, 204

- Microsoft Visual C/C++ compiler, 89
  - disassembly listing for switch statement, 402
  - vs. GNU g++, 161
  - and pack pragma, 137
- Microsoft Visual Studio, *dumpbin* utility, 25
- mIDA plug-in, 335, 492–494
- MIDL (Microsoft Interface Definition Language), 492
- MinEA function, 565
- MIPS
  - binary, script to mimic behavior, 275
  - processor, 48
    - IDA support for, 33
  - mitigation, of vulnerability, 465
- MK\_FP function, 564
- mkidp.exe* utility, 389–390
- mnemonics, 4
- modal dialog, 175*n*, 331*n*
- modeless dialog, 175*n*, 331*n*
- modifying data, IDC functions for, 259–260
- module* directory, 282, 366
- modules, 310
  - building using Unix-style tools, 283
- Modules window, 503
- mouse, support for console, 188, 189
- mov statement, 130, 272
- move\_seg function, 349–350
- MS-DOS header structure, for PE file, 153–154, 339–341
- MS-DOS stub, 389, 390
- MS-DOS.EXE loader, 46
- msg function, 299
- mutual ptrace, 437
- MZ tag, in MS-DOS executable file headers, 16*n*

## N

- nalt.hpp* file, and netnodes use, 288
- .nam* file, 49
- Name command, in IDA Sync, 486
- Name function, 262, 565
- NameChars option, 202
- named
  - constants, in source code, 114
  - licenses, 33
  - locations, changing, 105–107
  - program locations, maximum name length for, 202

- NameEx function, 263, 565
- name.hpp* file, 287
- names, 68
  - assigning to address of first instruction of node basic block, 185
  - decoration, 162
  - in disassemblies, 104–108
    - changing, 104
  - of functions, 117
  - mangling, 26, 162–163
  - register, 107–108
    - for trace log file, 509
- Names window, 55, 62, 68–69
  - adding name to, 106
  - in IDA Desktop, 56, 57
- NASM (Netwide Assembler), 9, 28
- navigation, 81
  - basics, 82–85
  - history, 84–85
    - Jump to address dialog, 84
    - in Linux console version, 190
- navigation band, 55, 57
- navigational targets, cross-references as, 83
- ndisam* utility, 310
- near address for flow, 169
- neighbors of function, 77
- netnode class, 288–289
- netnode.hpp* file, 287, 288, 289
- netnodenumber, 288, 291
- netnodes, 256, 288–296
  - creating, 289–291
  - data storage, 291–295
  - declaring, 290
  - deleting, 295–296
  - emulator state restored from, 446
  - and global arrays, 295
  - iterating over contents of array, 294–295
- Netwide Assembler (NASM), 9, 28
- network packets, shellcode in captures, 475–476
- new operator, 159, 160
- New option in welcome screen, 44–45
- New Projects dialog (Visual Studio), 320
- new\_til* function, 357
- NextAddr* function, 565
- NextFchunk* function, 565
- NextFuncFchunk* function, 565
- NextFunction* function, 263, 565
- NextHead* function, 565
- NextNotTail* function, 566

- NextSeg function, 566
- nm utility, 20–21
- nodeidx\_t operator, 290
- nodes in graph, 168
- non-Windows platforms, GUI interface
  - on, 196–197
- NOP slides, 473, 474, 476
- normal flow, 64*n*
- notifications
  - plug-ins access to debugger, 517
  - for plug-ins intercepting calls to processor, 393
  - for processor modules, 315, 385–386
  - unhooking, 316
  - of vulnerability to software maintainer, 465
- notify function, 385
- NOVICE option, 206
- NtContinue function, 542
  - conditional breakpoint on, 543
- ntdll.dll* file, 534, 536
- NtQueryInformationProcess function, 534–535
- NtSetInformationThread function, 536

**O**

- obfuscated code analysis, 417–455. *See also* de-obfuscation
  - anti–dynamic analysis techniques, 433–437
    - debugger detection, 435–436
    - debugging prevention, 436–437
    - instrumentation detection, 435
    - virtualization detection, 433–435
  - anti–static analysis techniques, 418–432
    - disassembly desynchronization, 418–421
    - dynamically computed target addresses, 421–427
    - imported function obfuscation, 428–432
    - targeted attacks on analysis tools, 432
- IDA debugger for, 525–544
- static de-obfuscation of binaries, 438–455
  - emulation-oriented
    - de-obfuscation, 443–455
  - scripted-oriented de-obfuscation, 438–443
- obfuscating obfuscators, 528
- obfuscation, 19, 50, 418
  - and compiler identification, 214
  - imported functions, 428–432
- objdump utility, 11, 23–24, 428
- object files
  - displaying information from, 23–24
  - nm utility to list symbols from, 20–21
- object-oriented concepts, 156
- objects
  - initializing, 161
  - life cycle, 160–161
- OEP (original entry point)
  - recognition, 522
- offset cross-references, 172–173
- o\_imm operand type, 374
- OllyDbg, 7, 522, 540
- OllyDump plug-in, 523
- o\_mem operand type, 374
- OpAlt function, 566
- OpBinary function, 566
- OpChr function, 566
- opcode bytes
  - display options, 113
  - obfuscation, 425–427
- Opcode Database, 472
- OPCODE\_BYTES option, in *ida.cfg* file, 202
- opcodes, 4
- OpDecimal function, 566
- OpenRCE.org, 35, 364
  - Anti Reverse Engineering Techniques Database, 436
  - download page, 479
- OpenSSL cryptographic library, 229
- OpEnumEx function, 566
- operating system
  - and local debugging, 500
  - and third-generation languages, 4
- operation codes, 4. *See also* opcodes
- OpHex function, 566
- OpHigh function, 566
- OpNot function, 566
- OpNumber function, 566
- OpOctal function, 566
- OpOff function, 566
- OpOffEx function, 566
- OpSeg function, 566
- OpSign function, 566
- OpStkvar function, 566
- OpStroffEx function, 566
- op\_t class, 297, 373–374
  - processor dependent fields in, 374

- optimization, 412*n*
- OPTION/ALT key, on Mac, 192–193
- Options menu
  - ▶ Colors, 207
  - ▶ Demangled Names, 162
  - ▶ Dump/Normal View, 189
  - ▶ Font, 58
  - ▶ General, 62, 65
  - ▶ Setup Data Types, 123
- ord function, 261, 571
- ordinal number, for exported function, 230*n*
- ordinary flows, 169, 170–171
- o\_reg operand type, 374
- original entry point (OEP)
  - recognition, 522
- OS X systems, 23
  - console specifics, 192–195
  - IDA debugger, 546–547
  - IDA version, installation, 36, 37
  - otool utility for, 24
  - Terminal Inspector dialog, 193
- otool utility, 23, 24
- out function, 380, 382
  - for Python processor, 383
- out.cpp* file, 380
- OutLine function, 381
- out\_line function, 381
- OutMnem function, 381
- out\_one\_operand function, 381
- outop function, 380, 381
  - for Python processor, 383–384
- output files, 241–245
- OutputDebugStringA function, 537
- outputter, in processor module, 366, 380–385
- out\_register function, 382
- out\_snprintf function, 381
- out\_symbol function, 381
- out\_tagoff function, 382
- out\_tagon function, 382
- OutValue function, 381
- overflow buffer, 100
- overloaded versions of functions, 162
- OverTheWire.org, Wargames
  - section, 477
- Overview Navigator, 55, 215, 427
- overview window, 64

## P

- Pack Database option, when closing file, 52
- pack pragma, 137, 151
- packed attribute, 137
- packet-analysis tools, 476
- padding bytes
  - for field alignment in structure, 145
  - to fill program section, 241
- PaiMei framework, 177
- panning, in Graph Overview window, 64, 184
- Parallels, 197
- parameters, for debugging process setup, 549
- Parameters option, for user xrefs
  - chart, 182
- ParseTypes function, 566
- parsing
  - errors, in IDC scripts, 255
  - header files, in C, 151–152
  - strings to populate type library, 357
  - structure declarations, in C, 150–151
- Pascal-style strings, 70
- password
  - for Ida Sync, 486
  - for remote IDA debugging, 549
- .pat* files, parsing, 221
- Patch bytes dialog, 238
- Patch Program menu, 238–241
- Patch word dialog, 239
- PatchByte function, 259, 566
- patch\_byte function, 298
- PatchDword function, 259, 566
- patches, 458
  - availability and application, 466
  - generation, 241–245
  - in *IDAPython* directory, 483
- patch\_long function, 298
- patch\_many\_bytes function, 298
- PatchWord function, 259, 566
- patch\_word function, 298
- pattern files, creating, 219–221
- pat.txt* file, 217, 220
- Pause command, in IDA debugger, 504
- PauseProcess function, 566
- pcap loader, 355–360
- pcap\_file\_header, 355
- pcf.exe*, 219
- PDB (Program Database) file, 50

- PE (Portable Executable) format, 8
- PE Editor utility, 19
- PE loader, 395
- PE Sniffer utility, 19
- PE Tools, 18–19
- PEB (process environment block), 445
- PEiD, 19–20
  - pe.ldw*, 46
  - pekf.exe*, 219
- percent symbol (%), in AT&T assembly syntax, 9
- permissions, for manually created sections, 346
- per-process basis, exception handling configured on, 539
- persistent data storage, in IDC scripts, 256–257
- pe\_scripts, 244
- pe.sig* file, 405
- piracy, Hex-Rays' stance on, 32
- plb.exe*, 219, 220
- plb.txt* file, 217, 220, 224
- Please choose a structure dialog, 152
- Please choose a symbol dialog, 114–115
- plug-ins. *See also* Hex-Rays; ida-x86emu (x86emu) plug-in
  - activating, 313
  - basics, 309–310, 479–480
  - building, 318–322
  - collabREate, 488–491
  - configuration file, 201–202
  - configuring, 323–324
  - for customizing processors, 393–395
  - event notification, 315–316
  - execution, 316–317
  - extending IDC, 324–327
  - IDA Sync, 485–488
  - IDAPython, 481–484
  - IDARub, 484–485
  - initializing, 313–315
  - installing, 322–323
  - life cycle, 312–313
  - mIDA, 335, 492–494
  - notification codes for intercepting calls to processor, 393
  - in remote debugging, 550
  - unhooking notifications, 316
  - user interface, 327–336
  - writing, 310–318
- PLUGIN object, 310
  - PLUGIN\_FIX flag, 312, 313–314
  - PLUGIN.init function, 312–313, 314
  - PLUGIN\_KEEP, 314
  - PLUGIN\_OK, 314
  - PLUGIN\_PROC flag, 314
  - PLUGIN.run function, 317
  - plugins* directory, 38
    - for SDK, 282
  - plugins.cfg* file, 323
  - plugin\_t class, 310
  - plugin\_t object, initializing, 312
  - PLUGIN.term function, 313, 314, 316
  - PLUGIN\_UNL flag bit, 313
  - .plw* file extension, 318
  - .plx* file extension, 318
  - .pmc* file extension, 318
  - PointerToRawData field, 343
  - polymorphism, 164
  - pomf166.exe*, 219
  - popa instruction, 442
  - popf instruction, 442
  - PopXML function, 566
  - port, for remote debugging, 549
  - Portable Executable (PE) format, 8
  - posterior lines, for comments, 110
  - PPC processor, IDA support for, 33
  - ppsx.exe*, 219
  - predefined comments, with loadint utilities, 234–236
  - preprocessor directive, to include *idc.idc* file, 254
  - preprocessor macros, 203
  - PrevAddr function, 566
  - PrevFchunk function, 566
  - PrevFunction function, 263, 566
  - PrevHead function, 566
  - Previous button, in welcome screen, 45–46
  - PrevNotTail function, 566
  - Print options, for user xrefs chart, 183
  - printf\_line function, 381
  - printing, stack frame contents, 304
  - private headers, objdump to display, 23
  - Problems window, 78
  - procedure linkage table, 460
  - procedures, 85. *See also* functions
  - process environment block (PEB), 445
  - Process Monitor, 435
  - Process Stalker component, 177

- processes. *See also* running processes
    - control in debugger, 504–511
      - breakpoints, 505–508
      - stack traces, 511
      - tracing, 508–510
      - watches, 511
    - image, for debugger, 523
    - launching under debugger control, 499–500
    - tracing, 437
  - processor flags, for changing conditional to absolute jumps, 420
  - processor modules, 38
    - analyzer in, 366, 371–376
    - architecture, 395–396
    - basics, 363–364
    - building, 389–392
    - comments for, 382–383
    - customizing existing, 393–395
    - emulator in, 366, 376–379
    - initializing, 370
    - mnemonic lookup for disassembly instruction, 369
    - notifications, 315, 385–386
    - options for, 203
    - outputter, 366, 380–385
    - post-processing, 391
    - specifying, 47
    - writing, 366–388
  - processor-dependent fields, in `insn_t` and `op_t` classes, 374
  - processors
    - adding predefined comments for, 235
    - IDA Pro support for, 33
  - `processor_t` struct, 366, 386
  - `processor_t.newasm` notification, 388
  - `processor_t.newprc` notification, 387
  - `procs` directory, 38
  - program addresses, assigning symbolic addresses to, 82
  - Program Database (PDB) file, 50
  - program entry point, 8*n*, 213. *See also* main function
    - listing into file, 73
    - relative virtual address (RVA) of, 342
  - program stack, emulated, placing data on top, 448
  - programming interface, 281
  - programming languages, 4
  - programs, in IDC scripts, 254–255
  - `pro.h` file, 285, 287
  - Project Selection dialog (CollabREate), 491
  - prologue, of function, 87
  - `ptmobj.exe`, 219
  - `ptrace`, 437
  - public name, 106
  - publish capabilities, in collabREate architecture, 489
  - PullThePlug.org, 477
  - pure virtual function, 158
  - purged bytes
    - for functions, 118
    - manually overriding, 231
  - Push All Function Names command, in IDA Sync, 486
  - push operations, 88, 92–93, 272
    - and function parameters, 130
  - pusha instruction, 442
  - pushf instruction, 442
  - PushXML function, 566
  - `.pyc` file extension, 364, 365
  - Python
    - byte code, 364–365
    - minimal analyzer for, 374–376
    - interpreter, 365
    - processor
      - emulator, 379
      - generated code from, 388
      - notify function, 386
      - `python_data` function, 384–385
- ## Q
- QuickEdit mode, 189
  - QuickUnpack, 426
- ## R
- radio buttons, on forms, 333
  - RCE (Reverse Code Engineering) forums, 35, 479
  - `rdtsc` instruction, 454–455
  - read cross-reference, 172
  - `readElf` utility, 24
  - reading data, IDC functions for, 259–260
  - `readlong` function, 262, 571
  - `readme.txt` file
    - of FLAIR tools, 217
    - of SDK, 366
  - `readshort` function, 262, 571
  - `readstr` function, 262, 571

- read/write breakpoints, 506
- read/write traces, 508
- RebuildImports.idc* script, 533
- Recent IDC scripts dialog, 250
- recently used files list, 45
- Recursive Depth option, for user xrefs
  - chart, 183
- recursive descent assembly, 11–14
  - disadvantage, 13
- Recursive option, for user xrefs
  - chart, 182
- redpill VMware-detection
  - technique, 434
- Refresh function, 566
- RefreshDebuggerMemory function, 566
- RefreshLists function, 566
- reg.cbp* file, 369
- register names, 107–108
  - as output, 382
- register\_extlang function, 574
- registers
  - accessing values from SDK, 519
  - displaying contents, 502
  - pointer to shellcode, 472
- register-to-memory transfer
  - instructions, 11
- registry key, 44
  - History subkey, 45
  - IDA option values, 207
- RegNames array, 369
- RegOpenKey function, disassembly of call
  - to, 228–229
- regparm keyword, in GNU gcc/g++, 91
- regular comments, 109
- regular expressions, in database
  - searches, 101
- relative virtual address (RVA), of program entry point, 342
- release binaries, vs. debug binaries, 412–414
- remote debugging, 547–550
- remote procedure call (RPC)
  - interface, 492
- RemoveFchunk function, 566
- RenameArray function, 257, 566
- RenameEntryPoint function, 566
- renimp.idc* script, 532
- repeatable comments, 109–110
  - customizing, 112
- reporting bugs, 58
- request\_COMMAND function, 517
- RET instruction (x86), 86, 89
- return instructions, 13–14
- return statement, in IDC scripts, 254
- Reverse Code Engineering (RCE)
  - forums, 35, 479
- reverse engineering, 5
  - Python byte code, 365
  - references for C++, 165–166
  - targeted attacks on tools, 432
- revert capability, IDA limitations, 53
- Rfirst function, 264, 567
- Rfirst0 function, 567
- RfirstB function, 264, 567
- RfirstB0 function, 567
- Rnext function, 264, 567
- Rnext0 function, 567
- RnextB function, 264, 567
- RnextB0 function, 567
- Roberts, J.C., 221
- Rolles, Rolf, 364
- rotate\_left function, 571
- RPC (remote procedure call)
  - interface, 492
- RTTI (Runtime Type Identification), 163–164
  - implementations, 404
- RTTICompleteObjectLocator structure, 164
- Ruby scripting, in IDARub, 484
- run function, in *plug-in\_t* class, 311
- Run To Cursor button (emulator), 447
- Run to Cursor command (IDA debugger), 505
- Run Until Return command (IDA debugger), 504
- running processes
  - analyzing, 18–19
  - attaching debuggers, 498–499
  - displaying list, 498
- RunPlugin function, 567
- runtime
  - computing address for execution flow at, 421
  - errors, in IDC scripts, 255
  - value, jump instruction target dependence on, 11
- Runtime Type Identification (RTTI), 163–164
  - implementations, 404
- RunTo function, 514, 567
- Rutkowska, Joanna, 434

## S

- Sabanal, Paul Vincent, 165
- SABRE Security, 186
- sandbox environments, 6, 426, 427, 523
  - for debugging malware, 500, 525
  - instrumentation, 435
- Save database dialog, 428
- Save disassembly desktop dialog, 209
- saved register value (“s”), 100
- saved registers, bytes for, 118
- saved return address (“r”), 100
  - vs. saved frame pointer, for variable offsets, 99
- savefile function, 262, 571
- save\_file function, 349
- save\_simpleton\_file function, 353–354
- scanning strings, 70
- ScreenEA function, 260, 567
- script cancellation dialog, 255
- scripted-oriented de-obfuscation, 438–443
- scripting, 204
  - to adjust import table entries, 532
  - associating with hotkeys, 258
  - basics, 249–250
  - disadvantages, 443
  - examples, 267–277
    - emulating assembly language behavior, 274–277
    - enumerating cross-references, 269–271
    - enumerating exported functions, 272
    - enumerating functions, 268
    - enumerating instructions, 268–269
    - finding and labeling function arguments, 272–274
  - execution, 250–251
  - IDA functions, 253–254, 258–267
    - for code cross-references, 264
    - for data cross-references, 265
    - data manipulation, 265–266
    - database names manipulation, 262–263
    - database search, 266–267
    - disassembly line components, 267
    - file input/output, 261–262
    - functions dealing with, 263
    - for reading and modifying data, 259–260
    - string manipulation, 261
    - user interaction, 260
  - IDC language, 251–257
    - error handling, 255–256
    - expressions, 252
    - persistent data storage, 256–257
    - programs, 254–255
    - statements, 252–253
    - variables, 251–252
  - to launch debugger and control
    - created process, 528–529
  - in remote debugging, 550
- SDK. *See* IDA Software Development Kit (SDK)
- sdk* directory, 36
- Search menu ▶ Next Sequence of Bytes, 102
- SEARCH\_CASE flag, 266
- SEARCH\_DOWN flag, 266
- search.hpp* file, 287
- searching
  - database, 100–102
    - text searches, 101
  - for structures, 153
- SEARCH\_NEXT flag, 266
- second-generation programming languages, 4
- section headers, objdump to display, 23
- SectionAlignment field, 342
- sections, 75
  - permissions for manually created, 346
- SegAddrng function, 567
- SegAlign function, 567
- SegBounds function, 567
- SegByBase function, 567
- SegByName function, 567
- SegClass function, 567
- SegComb function, 567
- SegCreate function, 567
- SegDefReg function, 567
- SegDelete function, 567
- SegEnd function, 567
- segment.hpp* file, 287, 343, 352, 362
- segments
  - creating in database, 343–344
  - editing dialog, 525
  - registers
    - access to configuration, 447
    - for IDA, 370
  - SDK functions for manipulating, 302–303
- Segments window, 75–76

- segment\_t class, 297, 302
- SegName function, 567
- SegRename function, 567
- SegStart function, 567
- SelectThread function, 567
- SelEnd function, 567
- self-modifying code, managing in static analysis environment, 438
- SelStart function, 567
- semaphore, 422*n*
- semicolon (;)
  - for comments, 108
  - for IDC statements, 252–253
- sequential flow instruction, 11
- server component
  - in CollabREate, 490–491
  - command line option, 548
  - in IDA Sync, 486
- SetArrayLong function, 257, 295, 567
- SetArrayString function, 257, 295, 568
- SetBmaskCmt function, 568
- SetBmaskName function, 568
- SetBptAttr function, 513, 568
- SetBptCnd function, 513, 568
- SetCharPrm function, 568
- SetColor function, 568
- SetConstCmt function, 568
- SetConstName function, 568
- SetDebuggerOptions function, 568
- SetEnumBf function, 568
- SetEnumCmt function, 568
- SetEnumFlag function, 568
- SetEnumIdx function, 568
- SetEnumName function, 568
- SetFchunkAttr function, 568
- SetFchunkOwner function, 568
- SetFixup function, 568
- SetFlags function, 568
- SetFunctionAttr function, 568
- SetFunctionCmt function, 568
- SetFunctionEnd function, 568
- SetFunctionFlags function, 568
- SetHashLong function, 568
- SetHashString function, 569
- SetHiddenArea function, 568
- set\_idc\_func function, 325
- SetLineNumber function, 569
- SetLocalType function, 569
- SetLongPrm function, 569
- SetManualInsn function, 568
- SetMemberComment function, 569
- SetMemberName function, 569
- SetMemberType function, 569
- set\_name function, 300
- set\_processor\_type function, 396
- SetProcessorType function, 569
- SetReg function, 569
- set\_reg\_val function, 519
- SetRegValue function, 512, 569
- SetRemoteDebugger function, 569
- SetSegmentAttr function, 569
- SetSegmentType function, 569
- set\_seg\_name function, 302
- SetSelector function, 569
- SetShortPrm function, 569
- SetSpDiff function, 569
- set\_start\_cs function, 571
- set\_start\_ip function, 572
- SetStatus function, 569
- SetStrucComment function, 569
- SetStrucIdx function, 569
- SetStrucName function, 569
- SetType function, 570
- Setup data types window, 123
- Setup strings window, 70
- SetXML function, 570
- sharing TIL files, 155–156
- shellcode
  - analysis, 475–477
  - register to point to, 472
- Shiva anti–reverse engineering tool, 426
  - IDA efforts to disassemble, 418–419
- Shiva ELF obfuscation tool for Linux, 437
- shnames array, 387
- SHOW\_SP option, in *ida.cfg* file, 202
- show\_wait\_box function, 317
- SHOW\_XREFS option, in *ida.cfg* file, 202
- shr instruction (x86), 441
- shrd instruction (x86), 441
- Siemens C166 microcontroller, reverse engineering binary image for, 339
- sig* directory, 38
- .sig* files, 214, 221
  - linking *.ids* file to, 233–234
- sigmake.exe* utility, 221
- sigmake.txt* file, 217, 221–222, 224
- signature files, 212
  - applying, 212–216
  - creating, 216–225
- Signature window, 76–77

- signatures
  - for identifying blocks of code, 76
  - startup, 224–225, 405
- signed elements, in array, 128
- signed shifts, 441
- simplex method, 231
- sizeof, for structure, 145
- Skape, 4, 430
- Skip command (emulator), 447
- Skochinsky, Igor, 165, 404
- Skoudis, Ed, 435
- snapshot, by CollabREate server, 491
- sockaddr structure, 74*n*
- SoftIce, 436
- software, interoperability, 7
- software breakpoints, 437, 505
- sort\_til function, 357
- SPARC processor, IDA support for, 33
- spoonm, 484
- sprintf function (CO), danger of, 270
- SQL database, 488
- SQLite database, 468
- .stack database segment, 445
- stack-allocated
  - arrays, 133–135
  - buffers, locating all functions with, 463–465
  - objects, destructors, 161
  - structures, 138–139
- stack-based virtual machine, 365
- stacks
  - bytes, for local variables, 118
  - distance between variables in, 470
  - frames, 67, 85–100
    - example, 91–95
    - IDA views of functions', 95
    - printing contents, 304
    - as specialized structures, 147
    - view, 99
  - manipulation operations, 11
  - pointers
    - adjustments, 120–121
    - advantage of using, 93
    - correcting computations, 121
    - customizing, 112
    - in Python processor, 379
    - register, emulator for report on behavior, 378
  - traces, in IDA debugger, 511
  - variables
    - changing names of, 97
    - formatting as structure, 148–149
    - renaming, 104–105
    - views, 95–100
- standard calling convention, 89
- standard file input/output variables,
  - SDK restricting access to, 285
- standard structures, 152–155
- \_start, vs. main function, 213
- start address, of functions, 117
- start entry point, 73
- StartDebugger function, 570
- startEA data member, 296
- Starting direction option, for user xrefs
  - chart, 182
- startup sequences, monitoring for
  - process, 500
- startup signatures, 224–225, 405
- statements, in IDC scripts, 252–253
- states, for plug-in loading, 312–313
- static
  - analysis, 6
    - techniques, 459
  - de-obfuscation of binaries, 438–455
  - emulation-oriented
    - de-obfuscation, 443–455
    - scripted-oriented de-obfuscation, 438–443
  - func attribute, 119
  - initialization of plug-ins, 313
  - linking, 22–23
  - variables, 72*n*
- static keyword, 253
- statically linked binaries, 178, 211
  - identifying and acquiring, 217–219
- stdcall calling convention, 120
  - and stack-pointer analysis, 231
- stdcall functions, 118
  - emulator and, 450
- \_stdcall modifier, 89
- Step Into command, in IDA
  - debugger, 504
- Step Over command, in IDA
  - debugger, 504
- StepInto function, 514, 570
- StepOver function, 514, 570
- StepUntilRet function, 514, 570
- StopDebugger function, 570
- store string byte (stosb) instruction, 441
- store\_type function, 358
- stosb (store string byte) instruction, 441
- strcpy function, 175, 459
  - danger of, 270

- stream disassemblers, 28–29
- strings, 124–126
  - in IDC scripts, 251, 252
  - for library identification, 218
  - manipulation, IDC functions for, 261
  - scanning, 70
  - variables, virtual repeatable
    - comment for, 110
- strings utility, 27–28
- Strings window, 55, 62, 70–71
  - in IDA Desktop, 56, 57
  - refreshing content, 442
- StringStp function, 570
- strip utility, 18
- stripping binary executable files, 18
- strlen function, 261, 572
- strstr function, 261, 572
- struc\_t class, 297, 301
- struct.hpp* file, 287
- structures
  - arrays of, 141–142
  - collapsing definition, 146
  - creating, 142–147
  - editing members, 145–146
  - enumerating members, 305–306
  - field offset, 145
  - formatting stack variables as, 148–149
  - globally allocated, 138
  - heap-allocated, 139–141
  - importance of correct layout, 152
  - importing, 150–152
  - manual layout, 143–147
  - member access, 136–142
  - notation for readability, 150
  - SDK functions for manipulating, 301
  - search for, 153
  - stack frames as specialized, 147
  - stack-allocated, 138–139
  - standard, 152–155
  - templates, 147–150
- Structures window, 74–75, 143
- subroutines, 85. *See also* functions
  - instruction calls to, 64*n*
- subscribe capabilities, in collabREate
  - architecture, 489
- substr function, 261, 572
- substring searches, 101
- subviews, in console versions, 189
- sub\_XXXXXX autogenerated names, 69
- summary tools, 20–26
  - c++filt utility, 25
  - dumpbin utility, 23, 25
    - for obfuscation, 428
  - ldd utility, 22–23
    - for obfuscation, 428
  - nm utility, 20–21
  - objdump utility, 11, 23–24, 428
  - otool utility, 23, 24
- summary view
  - context-sensitive menu, 97–98
  - for function’s stack frame, 95
- superclass constructors, 161
  - chain of calls to, 165
- superclass destructors, 161
- supset function, 293
- supstr function, 293
- supval function, 293
- supvals, 291, 292–294
- switch statement
  - for analyzer function, 372
  - for different compilers, 400–404
  - in function, 10
- symbol files, 38
- symbol-selection dialog, 114–115
- symbolic names, assignment to program addresses, 82
- symbols
  - objdump to display information, 24
  - removing from binary file, 18
- synchronous interaction, with
  - debugger, 519
- sysenter instruction, 91
- Sysinternals, Process Monitor, 435
- system call, 91

## T

- T, in nm utility output, 21
- t, in nm utility output, 21
- tabs, in IDA Desktop, 55
- Tabs function, 570
- tags, 291
- TailDepth function, 570
- tails, 296
- TakeMemorySnapshot function, 570
- tar archives, 36
- target addresses, dynamically
  - computed, 421–427
- targeted attacks on analysis tools, 432
- TASM (Borland Turbo Assembler), 9
- TCP port, for remote debugging, 549
- TEB (thread environment block), 445, 540–541

- tElock anti-reverse engineering tool, 422–424, 426, 430–431, 540
- templates, for structures, 147–150
- Tenable Network Security, 493
- Tenable Security, 335
- term function pointer, in `plug-in_t` class, 311
- terminating scripts, 255
- `term_output_butter` function, 381
- text searches, 101
- Text view, 66–67
  - switching between graph view and, 184
- third-generation programming languages, 4
- this pointer, 90
  - in C++ member functions, 156
  - returning in EAX register, 161
- thiscall calling convention, 90, 156
- thread environment block (TEB), 445, 540–541
- threads, IDA and, 317
- Threads window, 503
- thunk functions, 412–413
- ThunRTMain library function, 411
- til* directory, 39
- .til* files, 49, 77, 155–156, 228, 574
  - adding function prototype information to, 229
  - linking *.ids* file to, 233–234
- `til2idb` function, 356, 570
- TILIB utility, 574
- time stamp counter (TSC), 454
- tool tips, 55, 130*n*, 331
- toolbar area, 55
- toolbars
  - customizing, 208–209
  - for IDA debugger, 504–505
- top-level directory, for SDK, 282
- trace (single step) flag, detecting, 436
- tracing, in IDA debugger, 508–510
- Tracing Options dialog, 508–509
- TSC (time stamp counter), 454
- Turbo Assembler (TASM), 9
  - totuning.txt* file, 190
- type field, of operand, 374
- type libraries. *See also* *.til* files
  - populating by parsing string, 357
  - version 5.3 support for, 574
- Type Libraries window, 77
- typedef statement (C), 151

- TypeDescriptor structure, 164
- typeid operator, 164
- typeinf.hpp* file, 287, 357
- `type_info` structure, 164
- Types window, 155
- typinf.hpp* file, 356

## U

- U, in `nm` utility output, 21
- ua.hpp* file, 288, 381–382
- `ua_next_xxx` function, 372
- uncollapsing node, 186
- unconditional branching instructions, 11–12, 171
- unconditional jumps, in text view, 67
- `#undef` directive, 255
- Undefine command, 121–122
- undo, absence of, 39, 53, 61
- undocumented CPU instructions, 112–113
- Unicode strings, 70
  - search for, 101
- union, 144*n*
  - creating, 143–144
  - within `op_t`, 374
- universal unpacker, 530
- `unk_xxxxxx` autogenerated names, 69
- unpacking, automated, 528
- unsigned shifts, 441
- untar, 37
- upgrading
  - copying *idauser.cfg* file when, 203
  - IDA Pro, 34
- UPX, 426, 528
  - emulator to uncompress binary, 450
  - unpacking script for use with, 443
- USE\_DANGEROUS\_FUNCTIONS macro, 285
- user interaction
  - IDC functions for, 260
  - plug-in activation, 316
- user interface, 39
  - notifications, 315
  - for plug-ins, 327–336
  - SDK functions for, 299–300
- user xref charts, 181–183
- user-assigned names, characters allowed in, 202
- user-defined functions, 253
- users, for IDA Sync project, 487

*users.py* file (Ida Sync), 486  
USE\_STANDARD\_FILE\_FUNCTIONS  
    macro, 285, 354

## V

`var_` prefix for variable names, 97  
variables, 7  
    accessibility in IDC, 253  
    associating datatype with, 130  
    IDA name assignment, 97  
    in IDC scripts, 251–252  
    monitoring value during  
        debugging, 511  
    offsets, 94  
        generation, 99  
    standard file input/output, 285  
Veracode, 458–459  
versions of API  
    for loader module, 348  
    for plug-in, 310  
vertices in graph, 168  
VGA font for X server, 191  
View menu  
    ▶ Graphs  
        ▶ Flow Chart, 177  
        ▶ Function Calls, 179  
        ▶ User Xrefs Chart, 181  
        ▶ Xrefs From, 180  
        ▶ Xrefs To, 180  
    ▶ Open Subviews, 58  
        ▶ Cross-References, 174, 175  
        ▶ Disassembly, 66  
        ▶ Function Calls, 175  
        ▶ Imports, 532  
        ▶ Local Types, 150  
        ▶ Pseudocode, 480  
        ▶ Strings, 70  
        ▶ Type Libraries, 155  
virtual  
    addresses, 82  
        disassembly window  
            organized by, 84  
        finding useful, 473–475  
        names for, 68  
        in text view, 66  
    functions, 157–160, 173  
        machine, stack-based, 365  
        repeatable comment, for string  
            variables, 110  
VirtualAlloc function, emulated  
    version, 451

virtualization, detection, 433–435  
virtualization software, 197  
Visual Basic 60, startup code from compiled program, 411  
Visual C/C++ compiler, start routine for code, 410  
Visual C++ Express, for building IDA modules, 320  
Visual Studio 2005  
    for creating project, 320–322  
    plug-in configuration values, 322  
    program binaries from, 412  
VMware  
    Tools, 434  
        Windows registry entries for, 433  
    Workstations, 197  
Voids function, 570  
vtable pointer, 157, 159–160  
vtables, 157–160  
    and determining inheritance relationships, 165  
    layout, 158  
    polymorphic object pointer to, 164  
vulnerability advisory, 466  
vulnerability analysis, 6–7, 457–477  
    after-the-fact with IDA, 465–469  
    discovery with IDA, 458–465  
    exploit-development process, 469–475  
        finding useful virtual addresses, 473–475  
    instruction sequences location, 472–473  
    stack frame breakdown, 470–472  
    shellcode analysis, 475–477

## W

*.w32* file extension, 389  
wait function, 570  
wanted\_hotkey pointer, for plug-ins, 312, 324  
wanted\_name pointer, for plug-ins, 311, 324  
warning function, 260, 269, 299, 570  
watch lists, 511  
watches, in IDA debugger, 511  
watermark, for IDA copy, 32  
weak name, for named location, 107  
welcome message, 44  
wheel mouse, and graph zooming, 64  
Whittaker, Andy, 339

width, of form input field, 332

Win32 Application Wizard (Visual Studio), 321

*win32\_remote.exe*, 547

WinDbg debugger, 11

windows. *See* data displays

Windows installer file, 36

Windows key file. *See* *ida.key* file

Windows menu

- ▶ Load Desktop, 58, 209
- ▶ Remove/Move, 188
- ▶ Reset Desktop, 58, 209
- ▶ Save Desktop, 58, 209

Windows Multiple Document Interface (MDI), 331*n*

Windows operating system

- dumpbin* utility, 23
- IDA installation for, 36
- library handle, 451*n*
- Linux-style command shell for, 17
- obfuscated malware and, 450
- tool to de-obfuscate executables, 426

Windows PE file. *See also* Portable Executable (PE) format

- magic numbers to identify, 16
- manually loading, 339–347

Windows PE loader, 46

Windows processors, *.w32* file extension for, 389

Windows virtual machine, IDA

- within, 39

Wine, 39, 191, 197

- running *wingraph32* under, 192

*wingraph32* application, 176

WinHelp files, 204

WinLicense, 426, 431

Wireshark, 435, 476

word, 100

- changing in database, 239

word function, 259, 570

*word\_XXXXX* autogenerated names, 69

wrapper code, 179

write breakpoints, 506

write cross-references, 172

write four capability, 469*n*

write traces, 508

*writelong* function, 262, 572

*writeshort* function, 262, 572

*writestr* function, 262, 572

## X

X server, VGA font for, 191

X11

- for Wine on OS X, 197
- port of TVision libraries, 191
- Preferences dialog, 194

x86

- assembly language, formats for, 8–9
- binaries, generating pseudocode for functions, 480–481
- compilers, 87
  - fastcall convention, 89–90
  - RET instruction, 86, 89
- instruction set, stream disassemblers for, 28
- processor module, loaders for, 395

x86emu plug-in (*ida-x86emu*), 336, 444–445, 492

- additional features, 453–454
- and anti-debugging, 454–455
- breakpoints, 446
- control dialog, 445
- emulator-assisted de-obfuscation, 448–453
- functions emulated by, 451
- initializing, 445–446
- operation, 446–448

*.xinitrc* file, 194

*xmodmap* utility, 194

*xrefblk\_t* structure, 306–307

*xref.hpp* file, 288, 303

xrefs. *See* cross-references

Xrefs From graph, 181

Xrefs To graph, 180–181

*XrefShow* function, 570

*XrefType* function, 264, 265, 270, 570

*xtol* function, 261, 572

## Y

Yason, Mark Vincent, 165

*You may start to explore the input file right now* progress message, 57

## Z

Zbikowski, Mark, 16*n*

Zip files, for SDK, 280, 281

*zipids.exe* utility, 233

*ZwQueryInformationProcess* function, 535

Zynamics, 186, 467