[**Debugging**](http://arflstxtx.blogspot.com/2010/02/debugging.html)

**Debugging** adalah proses metodis menemukan dan mengurangi jumlah bug, atau cacat, dalam sebuah program komputer atau sepotong perangkat keras elektronik, sehingga membuatnya berperilaku seperti yang diharapkan. Debugging cenderung lebih sulit ketika berbagai subsistem yang erat digabungkan, seperti perubahan dalam satu dapat menyebabkan bug muncul di negara lain..  Karena melibatkan berbagai aspek, termasuk: interaktif debugging, control flow, integrasi pengujian, log file, pemantauan, memori dumps, Statistical Process Control, dan taktik untuk desain khusus meningkatkan deteksi sementara menyederhanakan perubahan.

**Asal**

Ada beberapa kontroversi mengenai asal-usul istilah "debugging."

Istilah "bug" dan "debugging" keduanya populer dikaitkan dengan Admiral Grace Hopper pada 1940-an.  Sementara ia bekerja pada Mark II Komputer di Harvard University, rekan-rekannya menemukan ngengat terjebak dalam sebuah relay dan dengan demikian menghambat operasi, dimana ia mengatakan bahwa mereka "debug" sistem. Namun istilah "bug" dalam arti kesalahan teknis tanggal kembali setidaknya ke 1878 dan Thomas Edison dan "debugging" tampaknya telah digunakan sebagai istilah dalam aeronautika sebelum memasuki dunia komputer. Memang, dalam sebuah wawancara Grace Hopper mengatakan bahwa dia tidak coining istilah.  Cocok ngengat terminologi yang sudah ada, jadi dia diselamatkan itu.The Oxford Dictionary inggris entri untuk "debug" mengutip istilah "debugging" acuan untuk digunakan dalam pengujian mesin pesawat dalam artikel tahun 1945 dalam Journal of the Royal Aeronautical Society, Hopper's bug itu ditemukan 9 September 1947. Seminalis artikel oleh Gill  Pada tahun 1951 adalah awal diskusi mendalam kesalahan pemrograman, tetapi tidak menggunakan istilah "bug" atau "debugging". . Dalam ACM 's perpustakaan digital, istilah "debug" digunakan pertama kali dalam tiga makalah dari 1.952 ACM Rapat Nasional. Dua dari tiga menggunakan istilah dalam tanda kutip.  Pada tahun 1963, "debug" adalah istilah yang cukup umum yang akan disebutkan secara sepintas tanpa penjelasan di halaman 1 dari CTSS manual. Artikel Kidwell *Stalking the Elusive Komputer Bug*  membahas etimologi dari "bug" dan "debug" secara lebih rinci.

**Ruang lingkup debug**

Seperti software dan sistem elektronik pada umumnya telah menjadi lebih kompleks, berbagai teknik debugging umum telah berkembang dengan metode-metode untuk mendeteksi anomali, menilai dampak, dan jadwal patch perangkat lunak atau penuh perubahan kepada sistem.  Kata-kata "anomali" dan "perbedaan" dapat digunakan, sebagai istilah yang lebih netral, untuk menghindari kata "kesalahan" dan "cacat" atau "bug" di mana mungkin ada implikasi bahwa semua yang disebut *kesalahan, cacat* atau *bug* harus tetap (di semua biaya). Sebaliknya, sebuah penilaian dampak dapat dibuat untuk menentukan apakah perubahan untuk menghapus sebuah *anomali* (atau *perbedaan)* akan biaya-efektif untuk sistem, atau mungkin yang dijadwalkan rilis baru mungkin akan membuat perubahan yang tidak perlu. Tidak semua masalah kehidupan-kritis atau misi-kritis dalam suatu sistem.  Juga, penting untuk menghindari situasi di mana perubahan mungkin akan lebih menjengkelkan bagi pengguna, jangka panjang, daripada hidup dengan masalah yang diketahui (di mana "mengobati akan lebih buruk daripada penyakit"). Mendasarkan keputusan dari penerimaan dari beberapa anomali dapat menghindari budaya dari "nol cacat" mandat, di mana orang mungkin tergoda untuk menyangkal keberadaan masalah sehingga hasilnya akan tampak sebagai nol *cacat.* Mempertimbangkan masalah-masalah jaminan, seperti biaya versus manfaat penilaian dampak, maka teknik-teknik debugging yang lebih luas akan memperluas untuk menentukan frekuensi anomali (seberapa sering sama "bug" terjadi) untuk membantu menilai dampaknya terhadap keseluruhan sistem.

**Peralatan**

 Debugging rentang, dalam kompleksitas, dari memperbaiki kesalahan sederhana untuk melakukan tugas-tugas yang panjang dan melelahkan pengumpulan data, analisis, dan penjadwalan update.  Debugging keterampilan para programmer dapat menjadi faktor utama dalam kemampuan untuk debug masalah, tetapi kesulitan debug software sangat bervariasi dengan kompleksitas sistem, dan juga tergantung, sampai batas tertentu, pada bahasa pemrograman  yang digunakan dan alat-alat yang tersedia, seperti *debugger.*  Debugger adalah perangkat lunak yang memungkinkan pemrogram untuk memantau pelaksanaan suatu program, hentikan, re-start itu, mengatur breakpoints, nilai-nilai perubahan dalam memori dan bahkan, dalam beberapa kasus, kembali ke masa lalu. Umumnya, tingkat tinggi bahasa pemrograman, seperti Java, membuat debug lebih mudah, karena mereka memiliki fitur seperti penanganan pengecualian yang membuat sumber-sumber nyata perilaku kacau lebih mudah untuk menemukan. Pada tingkat yang lebih rendah bahasa pemrograman seperti C atau perakitan, bug diam dapat menyebabkan masalah seperti korupsi memori, dan sering sulit untuk melihat di mana masalah awal terjadi. Dalam kasus-kasus, debugger memori alat mungkin diperlukan. Dalam situasi tertentu, tujuan umum perangkat lunak yang spesifik dalam bahasa alam dapat sangat berguna. Ini mengambil bentuk *alat analisis kode statis*. Alat ini mencari himpunan yang sangat spesifik masalah yang telah diketahui, beberapa yang umum dan langka, di dalam kode sumber. Semua isu-isu seperti itu terdeteksi oleh alat-alat ini akan jarang dijemput oleh kompilator atau penerjemah, sehingga mereka tidak sintaks catur, tapi lebih semantik dam. Beberapa alat mengklaim dapat mendeteksi masalah unik 300 +. Baik komersial dan alat gratis ada dalam berbagai bahasa. Peralatan ini dapat sangat berguna ketika memeriksa pohon-pohon sumber yang sangat besar, dimana hal itu tidak praktis untuk melakukan Walkthroughs kode. Contoh khas masalah akan terdeteksi variabel dereference yang terjadi *sebelum* variabel diberikan nilai. Contoh lain adalah dengan melakukan pengecekan tipe kuat ketika bahasa tidak memerlukan demikian. Dengan demikian, mereka lebih baik pada kemungkinan menemukan kesalahan, versus kesalahan aktual.  Akibatnya, alat ini memiliki reputasi positif palsu. The old Unix *lint* program is an early example. Unix tua *serat* program adalah sebuah contoh awal. Untuk debugging hardware elektronik (misalnya, perangkat keras komputer) maupun perangkat lunak tingkat rendah (misalnya, BIOS, device driver) dan firmware, instrumen seperti oscilloscopes, analisis logika atau di-circuit emulator (ICES) yang sering digunakan, sendirian atau dalam kombinasi. Sebuah ICE dapat melakukan banyak perangkat lunak khas tugas debugger pada tingkat rendah perangkat lunak dan firmware.

**Berbagai debug techiques**

Sering kali langkah pertama dalam debug adalah upaya untuk mereproduksi masalah. Ini bisa menjadi tugas yang tidak sepele, misalnya seperti proses paralel atau beberapa bug perangkat lunak yang tidak biasa. Juga, lingkungan pengguna spesifik dan penggunaan sejarah dapat membuat sulit untuk mereproduksi masalah. Setelah bug tersebut direproduksi, input dari program perlu disederhanakan untuk membuatnya lebih mudah untuk debug.Sebagai contoh, sebuah bug dalam kompilator dapat membuat crash ketika parsing beberapa file sumber besar.  Namun, setelah penyederhanaan kasus tes, hanya beberapa baris dari sumber asli file bisa cukup untuk mereproduksi kecelakaan yang sama. Penyederhanaan semacam itu dapat dibuat secara manual, menggunakan membagi-dan-menaklukkan pendekatan.Programmer akan mencoba untuk menghapus beberapa bagian dari pengujian awal kasus dan periksa apakah masalahnya masih ada.  Ketika debugging masalah dengan GUI, pemrogram dapat mencoba untuk melewati beberapa interaksi pengguna dari masalah asli deskripsi dan periksa apakah tindakan yang tersisa cukup untuk bug untuk muncul.  Kasus uji untuk mengotomatisasi penyederhanaan, *delta debugging*  metode dapat digunakan. Setelah kasus uji cukup sederhana, seorang programmer dapat menggunakan debugger program alat untuk memeriksa keadaan (nilai-nilai variabel, ditambah dengan panggilan stack) dan melacak asal-usul masalah. Dalam kasus sederhana, melacak hanya beberapa pernyataan cetak, yang output nilai-nilai variabel pada titik-titik tertentu pelaksanaan program. Remote debugging adalah proses debug sebuah program yang berjalan pada sistem yang berbeda daripada debugger. Untuk memulai debug terpencil, debugger terhubung ke sistem remote melalui jaringan. Setelah terhubung, debugger dapat mengendalikan pelaksanaan program pada sistem remote dan mengambil informasi tentang negara. Mayat debug adalah tindakan yang debug dump memori (atau core dump) dari suatu proses.  Tempat pembuangan sampah dari proses ruang dapat diperoleh secara otomatis oleh sistem, atau oleh seorang programmer-instruksi dimasukkan, atau secara manual oleh pengguna interaktif. Crash dump (core dump) sering dihasilkan setelah proses telah dihentikan karena adanya unhandled pengecualian.

**Anti-debugging**

Anti-debugging adalah "pelaksanaan satu atau lebih teknik dalam kode komputer yang menghambat upaya reverse engineering  atau debugging proses target".Jenis teknik ini adalah:

* API-based: memeriksa keberadaan debugger sistem menggunakan informasi.
* Exception-based: periksa untuk melihat apakah pengecualian yang menggangg.
* Proses dan thread blok: memeriksa apakah proses dan thread blok telah dimanipulasi.
* Memodifikasi kode: memeriksa kode modifikasi yang dilakukan oleh perangkat lunak penanganan debugger breakpoints.
* Hardware-dan mendaftarkan-based: cek hardware breakpoints dan CPU register.
* Timing dan latensi: mengecek waktu yang dibutuhkan untuk melaksanakan instruksi.

Debugging dapat dihambat dengan menggunakan satu atau lebih dari teknik-teknik di atas.  Ada cukup banyak teknik anti-debugging tersedia untuk perangkat lunak yang cukup melindungi terhadap sebagian besar ancaman.

[RSS](http://arflstxtx.blogspot.com/feeds/posts/default) Diposkan oleh arief di [21:37](http://arflstxtx.blogspot.com/2010/02/debugging.html) Label: