چرا نرم‌افزارها باگ دارند؟

اشاره :

یکی از مفاهیم بسیار جالب و در عین حال بسیار پیچیده در دنیای برنامه‌نویسی ، مفهوم باگ (Bug) یا نقص نرم‌افزاری است . واقعاً چرا نرم‌افزارها باگ دارند؟ چرا هیچ وقت شر این باگ‌ها از سرمان کم نمی‌شود؟

یک تلنگر فکری

خوب است گاهی از فضای روزمرگی بیرون بیاییم و به مفهوم واقعی کارهایی که انجام می‌دهیم و علل وقوع پدیده‌های اطراف خود نظری بیفکنیم و درباره آن‌ها دوباره بیندیشیم . یکی از مفاهیم بسیار جالب و در عین حال بسیار پیچیده در دنیای برنامه‌نویسی ، مفهوم باگ (Bug) یا نقص نرم‌افزاری است . شاید هیچ مفهوم و موضوع دیگری در علوم مهندسی را نتوان یافت که به اندازه مفهوم باگ ، این واقعیت مهم را برای انسان روشن کرده باشد که هیچ فرمول و قانون ساخت انسان ، بی‌اشکال و نقص نیست و در هر طرح و برنامه‌ای ، بدون تردید ، نقصان‌ها و لغزش‌هایی وجود دارد که در نگاه اول به نظر نرسیده است . بنابراین همواره باید در جهت اصلاح طرح‌ها ، برنامه‌ها ، قوانین و فرمول‌ها کوشید . به عقیده من موضوع باگ چندان مهم است که می‌توان درباره چرایی و چگونگی آن ساعت‌ها بحث کرد . امیدوارم در آینده نزدیک فرصتی فراهم آید تا در مورد ابعاد فلسفی این قضیه چیزی بنویسم ، اما در این مجال کوتاه می‌خواهم کمی درباره ابعاد علمی و مهندسی این موضوع گپ بزنم . واقعاً چرا نرم‌افزارها باگ دارند؟ چرا هیچ وقت شر این باگ‌ها از سرمان کم نمی‌شود؟

باگ چیست؟

همان طور که می‌دانید ، به هر نقص یا ایراد یک نرم‌افزار یا برنامه کامپیوتری باگ می‌گویند . باگ از نظر لغوی یعنی حشره کوچک و در تاریخ مهندسی نرم‌افزار گفته می‌شود این اصطلاح را اولین بار Grace Hoper ، خانمی که در دانشگاه هاروارد مشغول تحصیل و تحقیق در رشته کامپیوتر بود ، به کار برده است . او که در حال کار با کامپیوترهایMark II و Mark III بود ، یک‌بار با مشکل مواجه شد و تکنیسین‌هایی که برای بررسی مشکل و تعمیر کامپیوتر ، آن را باز کرده بودند سوسکی را پیدا کردند که وارد دستگاه شده بود و آن را از کار انداخته بود .

البته در حقیقت این واژه را اولین بار همان تکنیسین‌هایی که این حشره را داخل دستگاه یافته بودند ، طی یادداشتی (اولین دلیل واقعی باگ / ایراد برنامه پیدا شد) به شوخی به کاربرده بودند . البته این تکنیسین‌ها یا خانم هوپر اولین کسانی نبودند که از این واژه برای اشاره به یک ایراد در دستگاهی استفاده می‌کردند . آن‌ها صرفاً برای نخستین بار از این اصطلاح در دنیای کامپیوتر استفاده کردند ، ولی اعتقاد بر این است که اصطلاح Debugg توسط همین افراد ابداع شد .

موضوع باگ یکی از سرفصل‌های مهم رشته مهندسی نرم‌افزار است . از این رو متون و کتاب‌های مفصلی در زمینهDebugging یا اشکال‌زدایی از نرم‌افزار و متدهای آن تألیف شده است و همچنان ادامه دارد .

برنامه‌نویسان تازه‌کار معمولاً از این شاخه مهندسی نرم‌افزار گریزانند و امیدوارند برنامه‌هایی بنویسند که به قدری خوب باشد که اصلاً کارش به اشکال زدایی نکشد ، ولی پس از دو سه سال کار حرفه‌ای در این زمینه سرانجام تسلیم می‌شوند و آشنایی با اصول علمی اشکال زدایی برایشان به یک ضرورت تبدیل می‌شود ؛ مگر این‌که نخواهد به اصول اخلاقی و حرفه‌ای مهندسی نرم‌افزار متعهد باشند و از این‌که برنامه‌های ساخت آن‌ها پر از انواع باگ و ایراد باشد ، باکی نداشته باشند ، اما برطرف کردن باگ‌ها برای بسیاری از برنامه‌نویسان غیرآماتور یکی از قسمت‌های چالش برانگیز و لذت بخش کار است و تقریباً مثل حل کردن معما است .

کلیه حقوق مادی و معنوی این سایت متعلق به گروه شرکت های آرک می باشد

برنامه‌نویسانی که دائماً به فکر کاستن از باگ‌ها و ایرادهای نرم‌افزارهای خود هستند ، در حقیقت به طور مداوم در حال انجام یک ورزش فکری هستند که رشته‌ای تو در تو از حلقه‌های پرسش و پاسخ را در دل خود دارد .

با این همه ، اجازه بدهید به اختصار ببینیم اصولاً چرا باگ‌ها به وجود می‌آیند و ریشه این معضل کجاست؟

چرا باگ‌ها پدید می‌آیند؟

وقتی در دنیای سیستم‌های دیجیتالی و کامپیوتری از باگ صحبت می‌کنیم ، مقصودمان بیشتر یک نقص نرم‌افزاری است و کمتر پیش می‌آید یک نقص دیجیتالی سخت‌افزاری را باگ بنامیم . هرچند که این لغت از نظر تاریخی در مهندسی مکانیک و ادوات سخت‌افزاری ریشه دارد . بنابراین آن دسته از وسایل دیجیتالی که فاقد نرم‌افزارند ، اصولاً در این بحث جای نمی‌گیرند . مثلاً یک دستگاه ویدیو را با یک کامپیوتر مقایسه کنید . هر وقت که می‌خواهید ویدیو را روشن کنید ، یا روشن می‌شود یا نمی‌شود و حالت دیگری در این میان وجود ندارد ، اما وقتی یک کامپیوتر را روشن می‌کنید ، ممکن است سیستم عامل بالا نیاید (بوت نشود) . در این صورت می‌توان گفت کامپیوتر واقعاً به کار نیفتاده است ؛ زیرا با این که روشن است ، کار نمی‌کند . بنابراین ، هر مشکلی هست ، در ماهیت نرم‌افزاری باگ نهفته است .

در اینجا به پرسش اساسی‌تری می‌رسیم : اصلاً نرم‌افزار چیست؟

این پرسش کوچکی نیست و من هم در این یادداشت کوچک ادعای پاسخ دادن به آن را ندارم ، اما تا آنجا که به بحث ما مربوط می‌شود ، می‌توان گفت نرم‌افزار بسته‌ای حاوی یک یا چند دستورالعمل است که کسی (پردازنده) آن را در جایی (حافظه) اجرا می‌کند . پردازنده‌های امروزی به ندرت اشتباه می‌کنند . به علاوه، خود آن یک سخت‌افزار دیجیتالی است که بحث باگ چندان درباره عملکرد آن صدق نمی‌کند . در واقع اگر انصاف بدهید ، احتمال اشتباه محاسباتی پردازنده‌های امروزی اینتل و AMD در کامپیوترهای ما چنان ناچیز است که اصلاً قابل مطرح کردن نیست . پس باید بدانیم که این دستورالعمل کجا اجرا می‌شود؟ جواب <حافظه> است . در نتیجه پاسخ این معما هرچه باشد ، ماجرایی است که در ارتباط با دستورالعمل‌ها و حافظه رخ می‌دهد .

زنجیره عملیات

ساده‌ترین باگ نرم‌افزاری وقتی به وجود میآید که یک دستورالعمل غلط باشد یا از منطق نارسایی پیروی کند . مثلاً اغلب اوقات فرایندها به صورت مرکب اجرا می‌شوند : ابتدا کاربر عمل A را انجام می‌دهد ، سپس نرم‌افزار روتین B را اجرا می‌کند و آن‌گاه فرایند C اجرا می‌شود ، ولی فکرش را نکرده بودید اگر دفعتاً ابتدا B اجرا شود و سپس کاربر عمل A را انجام دهد و آن‌گاه C به اجرا گذاشته شود ، چه اتفاقی خواهد افتاد؟ این ممکن است به سادگی منجر به پیدایش باگ شود . افزودن قابلیت‌های جدید به نرم‌افزار نیز گاهی باگ‌های تازه‌ای را پدید می‌آورد . این باگ‌ها اغلب ناشی از ظاهر شدن فرایندهای جدیدی هستند که زنجیره عملیات یک دستورالعمل مرکب را پیچیده‌تر ، و توالی‌های غیرمجاز آن‌ها را بیشتر می‌کنند ، اما این تنها یک بعد قضیه است .

شرایط مرزی

یک رکن اساسی در پیدایش باگ‌ها موضوعی است که شرایط مرزی (Boundary Conditions) می‌نامیم . آیا تا به حال از خودتان پرسیده‌اید اگر یک روز با بالاترین سرعتی که سرعت‌سنج اتومبیل شما نشان می‌دهد برانید ، چه اتفاقی ممکن است بیفتد؟ شاید موتور ماشین منفجر شود! به این وضعیت خاص که بخشی از یک سیستم با شدیدترین ورودی‌ها (در اینجا پدال گاز و دنده خودرو) و شدیدترین خروجی‌ها (سرعت خودرو) روبه‌رو می‌شود ، اصطلاحاً <شرایط مرزی> یک سیستم می‌گویند .

نرم افزارها نیز سیستم هستند و به دلیل ماهیت خود ، به آسانی ممکن است در معرض شرایط مرزی قرار بگیرند و در چنان شرایطی رفتار نامتعادل و پیش‌بینی نشده‌ای از خود نشان دهند . مثلاً امتحان کنید ببینید اگر به جای تایپ کردن یک عدد

کلیه حقوق مادی و معنوی این سایت متعلق به گروه شرکت های آرک می باشد

معمولی در قسمتی از یک نرم‌افزار حسابداری ، حداکثر ارقام ممکنه (مثلاً 9 رقم) را در آن تایپ کنید و کاری کنید که در خودش ضرب شود ، چه اتفاقی می‌افتد؟ آیا سرریز می‌کند؟ اگر نرم‌افزار قاطی کرد ، این یک باگ است . دادن ورودی‌های غیرطبیعی به سیستم نیز ممکن است همین پیامدها را داشته باشد . بعید است هیچ انسان عاقلی در باک بنزین ماشینش چند لیتر مایع ظرف شویی بریزد ، اما در قسمتی از یک نرم‌افزار حسابداری می‌توان به جای تایپ یک عدد ، یک مگابایت Text به صورت copy-paste وارد کرد تا ببینیم آیا نرم‌افزار قاطی می‌کند یا خیر و این کاری است که هکرها خیلی به آن علاقه دارند!

حافظه ؛ شاه کلید باگ

اکنون این پرسش را می‌توان مطرح کرد که ماهیت شرایط مرزی در نرم‌افزارها چیست و چه عواملی این شرایط مرزی را تعیین می‌کنند؟

خوب که دقت کنید ، متوجه می‌شوید شرایط مرزی (نوع متغیرها و حجم ورودی قابل‌پذیرش توسط آن‌ها) موضوعی است که به حافظه مربوط می‌شود . در شرایط خیالی ، حافظه مورد استفاده نرم‌افزار شما بی‌نهایت است ، ولی در شرایط واقعی نرم‌افزارتان مرتباً به دیواری به نام <محدودیت‌های حافظه> برخورد می‌کند . بنابراین چه هنگامی که با مسئله شرایط مرزی روبه‌رو هستید و چه هنگامی که پردازنده کامپیوتر دستورالعمل نارسای شما را اجرا می‌کند ، منشأ پیدایش باگ‌ها ، اتفاقات پیش‌بینی نشده‌ای است که درون حافظه رخ می‌دهند . آنجا محل زد و خورد داده‌های قد و نیم‌قدی است که گاه بسیار کوچک‌تر از شرایط مرزی حافظه هستند و گاه با دیواره‌های آن برخورد می‌کنند . گاهی کوچکند ، اما سرشماری آن‌ها مثل شمارش کودکان در حال بازی در حیاط یک دبستان شلوغ ، کار سختی است .

دستورالعمل‌ها و متغیرها می‌آیند و می‌روند . بعضی پاک می‌شوند و بعضی همچنان در گوشه‌ای از حافظه می‌مانند و بایت‌هایی که می‌مانند ، شرایط مرزی را برای دستورالعمل‌ها و ورودی‌های بعدی دشوارتر می‌کنند . بامزه‌ترین نوع باگ‌ها در فرایندهای مرکب پدید می‌آیند . مثلاً روتین A را به حافظه می‌فرستید تا آنجا دنبال مقداری بگردد ؛ غافل از این‌که یک روتین دیگر قبلاً به صورت اتفاقی آنجا آمده و آن مقدار را پاک کرده است و حالا روتین A سرگردان این سو و آن سوی حافظه دنبال گمشده‌اش می‌گردد !

نتیجه گیری اخلاقی!

یک دیدگاه این است که علت اصلی به وجود آمدن باگ‌ها فضایی است که حافظه در اختیار دستورالعمل‌های برنامه ما می‌گذارد . از آنجایی که دقیقاً نمی‌دانیم این فضا برای برنامه ما کافی است یا خیر یا نمی‌توانیم تمام توالی‌های ممکن در فرایندهای مرکب را پیش‌بینی کنیم ، به راستی نمی‌دانیم درون حافظه چه اتفاقاتی در حال وقوع است .

سخت‌افزارهای دیجیتال معمولاً از چنین منطقی پیروی نمی‌کنند و دستورات در این‌گونه سیستم‌ها قوانین سفت و سختی را به اجرا می‌گذارند که مجال اندکی برای جولان دادن داده‌ها و دستورالعمل‌ها در یک فضای باز باقی می‌گذارند1. به همین دلیل ادوات سخت‌افزاری ، انعطاف‌پذیری و کارایی محدودی دارند . سیستم‌های نرم‌افزاری از فلسفه متفاوتی پیروی می‌کنند . آن‌ها به داده‌ها و دستورالعمل‌ها مجال می‌دهند با صور گوناگون با هم تعامل داشته باشند و در یک فضای نسبتاً باز جولان دهند و در عوض انعطاف پذیرترند و کارایی بیشتری دارند . منشأ خطا کردن نرم‌افزار همین است .

بنابراین به عنوان برنامه‌نویس اگر مایلید نرم‌افزارتان کمترین باگ را داشته باشد ، باید حافظه مورد استفاده نرم‌افزارتان را هنگام اجرا زیر نظر بگیرید و ببینید آنجا واقعاً چه خبر است . مدیریت حافظه بسیار مهم است ؛ نه فقط از این جهت که حافظه را از وجود متغیرهای بی‌استفاده و روتین‌های راکد پاک کنید ، بلکه از این جهت که نحوه تعامل ورودی‌ها ، خروجی‌ها و فرایندهای نرم‌افزار خود را به دقت مانیتور و تماشا کنید . حافظه کامپیوتر مانند آزمایشگاهی است که داده‌ها

کلیه حقوق مادی و معنوی این سایت متعلق به گروه شرکت های آرک می باشد

و دستورالعمل‌ها را در آن به جان هم می‌اندازید . پس خوب است همچون شیمیدانان به ارلن و بِشِر خود نگاه کنید و ببینید مولکول‌ها چگونه به هم واکنش نشان می‌دهند .

امیدوارم در آینده به بحث <اشکال‌زدایی از نرم‌افزار> بیشتر بپردازیم .

پی‌نوشت

1- مگر این‌که به firmware یا میان‌افزاری مجهز باشند که در این صورت احتمال وقوع اشتباه و خطا در کارشان بالا می‌رود .

کلیه حقوق مادی و معنوی این سایت متعلق به گروه شرکت های آرک می باشد